UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Julia Baldissera, Raphael Schwinden da Silveira

PROPOSTA DE UM MODELO PARA DETECÇÃO DE FRAUDES NA EMISSÃO DE CERTIFICADOS DIGITAIS NA ICP-BRASIL

Florianópolis

2017

Julia Baldissera, Raphael Schwinden da Silveira

PROPOSTA DE UM MODELO PARA DETECÇÃO DE FRAUDES NA EMISSÃO DE CERTIFICADOS DIGITAIS NA ICP-BRASIL

Trabalho de conclusão de curso submetida ao Curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Jean Everson Martina
Coorientador: Prof. Dr. Luis Otavio Campos Alvares

Florianópolis

2017

Julia Baldissera, Raphael Schwinden da Silveira

PROPOSTA DE UM MODELO PARA DETECÇÃO DE FRAUDES NA EMISSÃO DE CERTIFICADOS DIGITAIS NA ICP-BRASIL

Esta Trabalho de conclusão de curso foi julgada aprovada para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovada em sua forma final pelo Curso de Sistemas de Informação.

Florianópolis, 30 de outubro 2017.

Prof. Dr. Jean Everson Martina
Orientador

Banca Examinadora:

Prof. Dr. Luis Otavio Campos Alvares
Coorientador

Prof. Dr. Ricardo Felipe Custódio

Dedicamos este trabalho às nossas famílias, que sempre foram muito presentes e nos apoiaram durante toda essa trajetória. Também dedicamos aos nossos professores e amigos que auxiliaram da melhor forma possível.

AGRADECIMENTOS

Agradecemos a esta universidade, seu corpo docente, servidores e administração, que viabilizaram nosso crescimento pessoal e profissional. Também ao nosso orientador prof. Dr. Jean Everson Martina e coorientador prof. Dr. Luis Otavio Campos Alvares, pelos incentivos e correções necessárias. Este agradecimento também se estende a todos que fizeram parte da nossa formação de alguma forma.

RESUMO

A assinatura e certificação digital são mecanismos eficientes para garantia de autenticidade, integridade e confidencialidade de documentos eletrônicos. Uma vez que estes mecanismos vêm tornando-se cada vez mais difundidos no meio eletrônico, o mesmo vem ocorrendo com as fraudes. Os órgãos emissores de certificados digitais têm, em seu fluxo de emissão, a verificação dos dados referentes ao pedido para garantir de que não se trate de uma fraude. Este processo, no entanto, é feito manualmente e, por isso, está sujeito a falhas por erro humano. Por conta disto, é proposto um modelo de detecção de fraudes que faça uso de técnicas de *Data Mining* a ser inserido no fluxo de emissão de certificados digitais, de forma que o processo torne-se mais preciso e mais ágil. Para isso, foram realizados estudos sobre os dados disponíveis, bem como estudos das técnicas de *Data Mining* mais adequadas ao cenário. Um modelo de classificação de dados foi implementado e testado no fluxo de um sistema fictício, confirmando sua viabilidade e contribuição com a agilidade do processo.

Palavras-chave: Mineração de Dados, Certificado Digital, ICP-Brasil.

ABSTRACT

Digital signature and certification are efficient mechanisms to ensure authenticity, integrity and confidentiality of electronic documents. Since these mechanisms become more and more widespread among the eletronic environment, the same has been occurring with frauds. The digital certificate issuing entities already have, in their issuing flow, verification of the data related to the request to ensure that it is not a fraud. This process, however, is done manually and is therefore subject to failures due to human error. Because of this, it is proposed a fraud detection model that makes use of Data Mining techniques to be inserted in the flow of digital certificates issuance, so that the process becomes quicker and more accurate. To achieve this, studies were carried out on the available data, as well as studies of Data Mining techniques that best fit the scenario. A data classification model was implemented and tested in the flow of a fictitious system, confirming its feasibility and contribution with the agility of the process.

Keywords: Data Mining, Digital Certificate, PKI-Brazil

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Com o grande avanço da tecnologia no cotidiano das pessoas, muitas ações e transações antes efetuadas presencialmente, passam a ser feitas no meio eletrônico pela internet. Tornou-se possível fazer declarações para a receita federal, emitir notas fiscais, gerar declarações de serviços médicos, fornecer dados para o Sistema de Recolhimento do FGTS (SEFIP), fornecer informações à previdência social, entre outras tarefas (CERTISIGN, 2017).

A assinatura e certificação digital aparecem nesse cenário para facilitar e garantir a autenticidade, integridade e confidencialidade desses documentos eletrônicos (ADAMS; LLOYD, 2003). Pessoas e organizações podem criar o certificado em uma autoridade confiável, e este servirá como uma identidade, pois contém informações do proprietário e servirá para efetuar as assinaturas dos documentos.

O modelo adotado no Brasil para gerenciar certificados digitais é a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil). Composta basicamente por Autoridades Certificadoras (ACs) e Autoridades de Registro (ARs), trabalhando em conformidade com as diretrizes e normas técnicas definidas pela AC Raiz (ITI, 2017b).

Apesar de todas as fiscalizações e especificações técnicas existentes, são noticiados vários casos de fraudes e incidentes com certificados digitais, como o grande esquema de saques irregulares de FGTS (BRASIL, 2017a). As consequências destes incidentes podem ser graves, já que documentos eletrônicos são utilizados nas mais diversas áreas da sociedade.

Dito isto, este trabalho apresenta uma proposta de modelo para detecção de fraudes na emissão de certificados digitais na ICP-Brasil. Este modelo é baseado em descoberta de conhecimento usando técnicas de *Data Mining* em cima dos dados existentes. Assim podem ser identificadas irregularidades, que comumente passam despercebidas pelos indivíduos envolvidos.

A metodologia utilizada no desenvolvimento deste trabalho tem caráter exploratório e aplicado, onde fez-se o uso de estudos relacionados às áreas de certificação digital e ciência de dados para elaboração de modelos de *Data Mining* mais precisos e eficientes, e inserção deste no fluxo atual de emissão de certificados digitais.

1.1 JUSTIFICATIVA

O Portal CD Brasil (BRASIL, 2017b) relatou a emissão de mais de 14 milhões de certificados digitais pelo Instituto Nacional de Tecnologia da Informação até 2015, sendo que, destes, mais de 3 milhões de certificados foram emitidos em 2015. O mercado de certificado digital mostra uma clara tendência de crescimento.

Desde junho de 2016, as autoridades da ICP-Brasil passaram a ser obrigadas a comunicar tentativas de fraude e irregularidades na emissão de certificados digitais com o objetivo de popular um banco de dados do novo sistema de comunicado de tentativa de fraude adotado pelo ITI, que oferece uma Lista Negativa para as ARs (ITI, 2017d). Essa lista contém as informações dos supostos fraudadores, incluindo características físicas, imagem do documento de identificação utilizado, entre outros dados. O Agente de Registro (AGR) fica responsável por fazer pesquisas para validar se a pessoa que está no processo de emissão de certificado digital não está envolvida em tentativas de fraudes anteriores.

O modelo de comunicado de fraudes acima mostra-se operacionalmente demorado, considerando a tendência de crescimento do mercado e o aumento da quantia de informações. Muitas vezes decisões são tomadas por intuição de indivíduos, sem levar em consideração o conhecimento incorporado nestes dados (PEI; HAN; KAMBER, 2012). Aqui entra a necessidade de ferramentas e técnicas poderosas para análise dos dados, as quais *Data Mining* fornece, que possibilitem uma predição de possíveis fraudes, conforme mencionado anteriormente.

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

Definir um modelo para predição de possíveis fraudes no processo de emissão de certificados digitais da ICP-Brasil, que receba informações das solicitações de certificados digitais, explore esses dados através de um projeto de *Data Mining* e retorne a classificação para novas entradas. Após isso, simular a utilização do modelo de detecção de fraude integrado ao Sistema de uma Autoridade Registradora.

1.2.2 Objetivos Específicos

- Entender o processo de solicitação e emissão de certificados digitais na ICP-Brasil;
- Desenvolver um projeto de *Data Mining*;
- Elencar modelos para o sistema de detecção de fraudes na estrutura da ICP-Brasil;
- Implementar um *web service* REST para receber informações e fornecer a classificação de novas entradas;
- Desenvolver um protótipo de sistema de validação presencial de uma AR;
- Elaborar um manual com instruções para a integração do serviço;
- Simular o funcionamento do modelo.

1.3 LIMITAÇÕES

A limitação deste trabalho é obter acesso aos dados reais sobre fraudes de certificados digitais. Esses dados estão armazenados no Sistema de Fraudes atualmente utilizado pelo Instituto Nacional de Tecnologia da Informação (ITI), que é a autarquia federal responsável pela ICP-Brasil.

Apesar da Constituição permitir o acesso a informação de órgãos públicos, previsto no inciso XXXIII do art. 5o (BRASIL, 1988), existem ressalvas a dados que sejam sigilosos. A Lei Nº 12.527/2011 regula o acesso a essas informações (BRASIL, 2011) e classifica informação sigilosa aquela que compromete atividades de investigação ou fiscalização em andamento. Logo não foi possível obter dados reais do contexto investigado.

1.4 METODOLOGIA

Do ponto de vista da natureza da pesquisa, a metodologia utilizada para este trabalho foi uma pesquisa aplicada. Isso devido a finalidade de produzir conhecimento para uma aplicação prática, com interesses pontuais (PRODANOV; FREITAS, 2013). Neste caso, foi produzido conhecimento a respeito do processo de emissão de certificados

digitais na ICP-Brasil, mais especificamente a parte de validação presencial do requerente do certificado.

Do ponto de vista dos objetivos, a pesquisa pode ser considerada exploratória, porque houve o propósito de produzir mais informações sobre o assunto a ser investigado (PRODANOV; FREITAS, 2013). Foram geradas mais informações a respeito de detecção de fraudes em um novo contexto: ICP-Brasil.

Inicialmente, são revisados conceitos teóricos fundamentais encontrados na bibliografia, para entender melhor conceitos relacionados a certificados digitais, infraestrutura de chaves públicas (especialmente a ICP-Brasil), descoberta de conhecimento e *Data Mining*. Em seguida, é realizado o estudo de trabalhos correlatos a respeito de detecção de fraudes em outros contextos da sociedade, como transações de cartão de crédito, telecomunicações, etc.

Com o embasamento teórico, foi possível realizar o processo de Data Mining com as informações obtidas na etapa de validação dos dados do requerente do certificado. Também foi possível criar um modelo de detecção de fraudes no processo de emissão de um certificado digital ICP-Brasil, considerando a integração deste serviço na infraestrutura existente.

1.5 ORGANIZAÇÃO DOS CAPÍTULOS

O trabalho foi organizado em quatro capítulos. O primeiro capítulo, "Introdução", faz uma apresentação das justificativas, objetivos do trabalho, limitações identificadas e metodologia utilizada.

O segundo capítulo, "Revisão Bibliográfica", traz a fundamentação teórica. Foram feitas revisões de livros, artigos e conteúdo da internet, sobre assuntos relacionados ao foco desta pesquisa. O terceiro capítulo aborda os trabalhos correlatos sobre detecções de fraudes em outros contextos da sociedade.

O quarto capítulo, "Proposta de Detecção de Fraudes", descreve os possíveis modelos definidos para a proposta, qual foi escolhido, seu desenvolvimento e a simulação da sua utilização. Também são detalhados os problemas enfrentados. No último capítulo é apresentada a conclusão do trabalho desenvolvido e os trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 CRIPTOGRAFIA

Compreender criptografia é extremamente importante para entender o que é um certificado digital e seu funcionamento. Criptografia (do grego "escrita escondida") é o emprego de técnicas para codificar informação (normalmente escrita em texto puro) em uma sequência de caracteres que não façam sentido para um leitor que não souber como decodificá-lo (RITTER, 2007). Para criptografar uma dada mensagem são necessárias duas entradas de dados: a mensagem em si e um código secreto tipicamente chamado de **chave criptográfica** ou, apenas, **chave** (HOUSLEY; POLK, 2001).

Existem relatos de uso da criptografia desde as civilizações egípcias há aproximadamente 4 mil anos (KAHN, 1967), ou seja, a codificação de mensagens não é nenhuma ideia moderna ou novidade para a humanidade. Entretanto a necessidade do uso da criptografia na área da segurança da informação surgiu com o advento do uso do computador e de sistemas de comunicação em meados de 1960 (MENEZES; OORSCHOT; VANSTONE, 1996).

2.1.1 Criptografia Simétrica

Um método de criptografia é a criptografia simétrica. A criptografia simétrica é o modelo de criptografia no qual tanto o emissor quanto o receptor da mensagem utilizam chaves criptográficas de mesmo valor. Por esse motivo, sistemas que utilizam chaves simétricas são comumente chamados de **sistemas de chaves compartilhadas** (HOUSLEY; POLK, 2001).

A criptografia simétrica deixa um pouco a desejar no quesito de segurança no gerenciamento das chaves, uma vez que qualquer receptor, em posse da mesma chave, pode decodificar uma mensagem, mesmo que esta não seja endereçada a ele (HOUSLEY; POLK, 2001).

2.1.2 Criptografia Assimétrica

A criptografia assimétrica (ou criptografia de chave pública) é o modelo de troca de informação criptografada que utiliza duas chaves

complementares, uma chave para codificar a informação (também chamada de **chave privada**) e outra para decodificar (chamada de **chave pública**). Desta forma, a chave pública pode ser distribuída, enquanto a chave privada deve ser armazenada em segurança. Apesar das chaves trabalharem de forma complementar, o valor da chave privada não pode ser determinado sabendo-se o valor da chave pública (MENEZES; OORSCHOT; VANSTONE, 1996; RITTER, 2007).

O gerenciamento de chaves ocorre de forma mais simplificada na criptografia assimétrica devido à significante redução da quantidade de chaves que devem ser mantidas em segredo (HOUSLEY; POLK, 2001).

2.1.3 Resumo Criptográfico

Para que o conceito de Resumo Criptográfico seja compreendido, antes é preciso falar sobre **funções *hash***. As funções *hash* são operações computacionais que convertem dados de tamanho arbitrário para uma sequência de caracteres de tamanho fixo (RITTER, 2007).

Um resumo criptográfico (ou resumo) é o resultado da execução de uma função *hash* sobre a mensagem, de forma que a mensagem emitida tenha um código que "represente"seu valor, mas em termos muito mais viáveis de se trabalhar do ponto de vista computacional. Outra característica é que esta função é unidirecional, e não permite descobrir o conteúdo a partir do código resultante (RITTER, 2007).

Em contraste com a facilidade de processamento computacional, idealmente, as funções *hash* usadas para autenticação devem ser "fortes", de modo que seja "computacionalmente inviável"encontrar outra mensagem que corresponda a um dado resumo criptográfico que não seja a mensagem que já se tem (RITTER, 2007).

2.2 ASSINATURA DIGITAL

A assinatura digital de documentos consiste no anexo de um código à mensagem, por parte do emissor, que dê ao receptor desta a certeza de que esta foi escrita por quem a confirma ter emitido e por mais ninguém (HOUSLEY; POLK, 2001). A assinatura de um documento dá-se pela seguinte sequência de passos:

- No lado do emissor:
    1. É calculado um resumo da mensagem a ser emitida utilizando uma função *hash* qualquer;

2. O resumo é criptografado utilizando a chave privada do emissor;
3. O resumo criptografado é anexado à mensagem como assinatura.

- No lado do receptor:

1. É calculado novamente um resumo da mensagem recebida com a mesma função *hash* utilizada na geração do resumo pelo emissor;
2. A assinatura da mensagem é decodificada com a chave pública do emissor, resultando num resumo decodificado;
3. Os resumos calculado e decodificado são comparados. Caso sejam iguais, tem-se a garantia de que a mensagem não foi adulterada.

2.3 CERTIFICADO DIGITAL

Um problema básico associado à assinatura digital é a falta de garantia de que o emissor de uma mensagem assinada digitalmente é quem ele diz ser. Mesmo o emissor sendo o detentor da chave privada utilizada na geração da assinatura. Os certificados digitais são a solução para isso.

Um certificado digital é tido como o elemento mais básico de uma Infraestrutura de Chaves Públicas (ICP). É um componente contendo, além da chave pública, informações básicas de seu detentor (nome, local de trabalho, dados de contato, etc.), algo que permita a um receptor associar o certificado à pessoa que se diz emissora da mensagem (HOUSLEY; POLK, 2001).

Apesar da eficácia do certificado descrito acima, pode haver um momento em que as informações descritas nele estejam desatualizadas. A solução para este problema está no uso da Infraestrutura de Chaves Públicas para garantir a veracidade das informações. É mais fácil para um receptor confiar no emissor da mensagem assinada digitalmente se houver a garantia de um órgão regulador de que um certificado digital é válido e que as informações nele contidas são verdadeiras.

Com base nisso, Housley e Polk (2001) descreve um "certificado ideal"como um elemento composto pelas seguintes propriedades:

1. Seria um objeto completamente digital, de modo que possa ser distribuído pela internet e processado de forma automática;

2. Conteria o nome do usuário detentor de sua chave privada, informações de contato e identificaria a organização onde este trabalha;

3. Seria fácil saber se o certificado foi publicado recentemente;

4. Seria criado por uma parte confiável (uma ICP, por exemplo) ao invés do próprio usuário detentor da chave privada;

5. Uma vez que a parte confiável poderia emitir mais de um certificado por usuário, deveria ser fácil diferenciar os certificados entre si;

6. Seria fácil identificar se o certificado é genuíno ou forjado;

7. Seria inviolável de forma que ninguém pudesse alterar seu conteúdo;

8. Os dados devem ser facilmente identificados como desatualizados, caso estejam;

9. Deve ser facilmente identificadas as aplicações para que o certificado pode ser usado.

2.4 AUTENTICAÇÃO MÚTUA

Stallings (2008) descreve Autenticação Mútua como um protocolo de autenticação que dá garantia de identidade a ambas as partes da comunicação. Ou seja, um emissor tem certeza de quem é o receptor da mensagem e vice-versa. Esta garantia é feita utilizando-se de técnicas de assinatura digital (Seção 2.2) para criptografar as mensagens trocadas.

2.5 INFRAESTRUTURA DE CHAVES PÚBLICAS DO BRASIL

Após compreender o que é certificado digital e a sua utilidade, é necessário entender como distribuí-los de forma segura e confiável. Uma Infraestrutura de Chaves Públicas (ICP) é um órgão que tipicamente contém a estrutura, políticas e padrões para gerenciar a criação, distribuição e revogação de certificados digitais (ADAMS; LLOYD, 2003).

A ICP-Brasil foi criada com base no modelo de ICP hierárquica, e foi instituída de acordo com a Medida provisória 2.200-2 de 24 de agosto

de 2001 para garantir autenticidade, integridade e validade jurídica de documentos eletrônicos (BRASIL, 2001). É composta por: um Comitê Gestor, uma AC raiz, ACs e ARs.

O Comitê Gestor tem autoridade para gerir políticas a serem executadas pela AC Raiz. Esse comitê tem representantes das partes interessadas, representantes da sociedade civil, e representantes de alguns órgãos como: Ministério da Justiça; Ministério da Fazenda; Ministério do Desenvolvimento, Indústria e Comércio Exterior; Ministério do Planejamento, Orçamento e Gestão; Ministério da Ciência e Tecnologia; Casa Civil da Presidência da República e Gabinete de Segurança Institucional da Presidência da República (BRASIL, 2001).

A AC Raiz tem a maior autoridade e está no topo da hierarquia de certificação. Cabe a essa executar as políticas, normas técnicas e operacionais impostas pelo comitê gestor. Segundo a Medida provisória 2.200-2 (BRASIL, 2001) a AC Raiz tem a função de emitir, distribuir, revogar e gerenciar certificados das ACs imediatamente subsequentes, assim como executar atividades de fiscalização e auditoria.

As ACs no geral são entidades autorizadas a emitir certificados digitais e vincular determinado titular ao par de chaves criptográficas. Essas têm responsabilidade por emitir, expedir, distribuir, revogar e gerenciar os certificados; além de disponibilizar aos usuários listas de certificados revogados, assim como outras informações válidas, mantendo registro de suas operações (BRASIL, 2001).

As ARs são entidades vinculadas a determinadas ACs. Cabe às ARs identificar e cadastrar usuários que devem comparecer presencialmente, além de encaminhar solicitações de certificados às ACs e manter registros de suas operações (BRASIL, 2001).

Os componentes da ICP-Brasil são ilustrados na Figura 1.

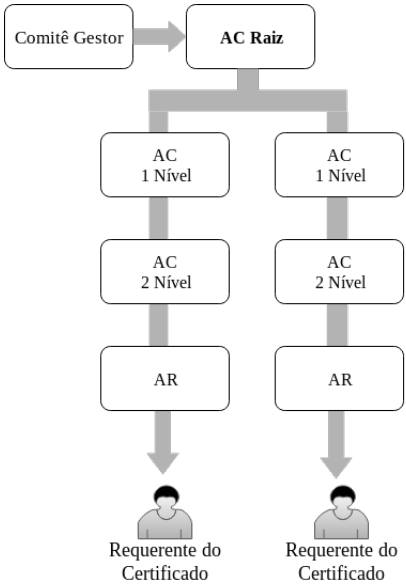

Figura 1 – Estrutura básica ICP-Brasil

2.6 INSTITUTO NACIONAL DE TECNOLOGIA DA INFORMAÇÃO

A AC Raiz da ICP-Brasil é o ITI. Este é reconhecido como autarquia federal ligada a Casa Civil da Presidência da República (BRASIL, 2001), e tem como missão:

> Atuar na inovação, regulação e provimento de soluções tecnológicas que garantam segurança, autenticidade, integridade, privacidade e validade jurídica de documentos e transações eletrônicas, respeitando o cidadão, a sociedade e o meio ambiente (ITI, 2017c).

O ITI é o órgão público responsável por manter e auditar a ICP-Brasil. Entre as atribuições descritas no site oficial (ITI, 2017c), o ITI deve executar as normas técnicas e operacionais, e Políticas de Certificados definidas pelo Comitê Gestor.

Outra atribuição é a de garantir a integridade, a autenticidade e a validade jurídica de documentos eletrônicos das aplicações de suporte e habilitadas que usufruem de certificados digitais da ICP-Brasil, assim

como a efetuação segura de transações eletrônicas.

O ITI também deve atualizar, revisar e, se necessário, ajustar os procedimentos e as práticas definidas para a ICP-Brasil. Também tem a atribuição de garantir a compatibilidade e fomentar a atualização das tecnologias do sistema e a adequação com as políticas de segurança (ITI, 2017c). No exercício de suas ocupações, o ITI desempenhará atividade de fiscalização, podendo ainda aplicar sanções e penalidades, na forma da lei (BRASIL, 2001).

Atribui-se ao ITI, também, incentivar projetos de desenvolvimento tecnológico e de pesquisa científica com objetivo de aumentar a cidadania digital, a inclusão digital e a popularização de certificados digitais. Deve atuar em cima de sistemas criptográficos, software livre, hardwares adeptos a padrões abertos, entre outras atribuições (ITI, 2017c).

2.6.1 Emissão de um certificado digital ICP-Brasil

Para obter um certificado, é necessário escolher uma Autoridade Certificadora (AC) credenciada pela ICP-Brasil, fazer a solicitação no próprio sistema da AC para a emissão do certificado digital, configurando-o como melhor lhe convir.

O certificado pode ser para uma pessoa física ou jurídica. Os certificados possuem políticas, como dito anteriormente, que são estabelecidas pelo ITI (2017e). No Documento "Requisitos mínimos para as políticas de certificado na ICP-Brasil", são descritos os tipos de certificados existentes:

- Tipos A1, A2, A3 e A4 servem para confirmação de identidade e assinatura de documentos eletrônicos com verificação da integridade de suas informações.

- Tipos S1, S2, S3 e S4 servem para cifração de documentos, bases de dados, mensagens e outras informações eletrônicas, com o objetivo de garantia de sigilo.

- Tipos T3 e T4 são utilizados em aplicações mantidas por autoridades de carimbo do tempo credenciadas na ICP-Brasil, para assinatura de carimbos do tempo.

- Tipos A CF-e-SAT são utilizados exclusivamente em equipamentos para assinatura de Cupom Fiscal Eletrônico – CF-e por meio

do Sistema de Autenticação e Transmissão de Cupom Fiscal Eletrônico – SAT.

Os tipos de certificados mais populares são o tipo A1, com validade de um ano e armazenado no computador, e o tipo A3 com validade de até cinco anos e armazenado em cartão ou *token*. De acordo com cada AC, existem informações sobre em que aplicações utilizar o certificado, os custos envolvidos, formas de pagamento, equipamentos, documentos necessários e outras exigências (ITI, 2017a).

Uma etapa importante do processo de emissão é a validação presencial, na qual o requerente vai pessoalmente a uma AR vinculada à AC escolhida para validar os dados preenchidos na solicitação anteriormente feita. Para isso é feito o agendamento diretamente com a AR que informará ao solicitante quais os documentos necessários (ITI, 2017a). Um AGR fica responsável pela execução desta verificação e validação. Após isso, caso não seja encontrada nenhuma irregularidade, o certificado é entregue ao cliente.

Essas etapas podem ser observadas na Figura 2, a seguir.

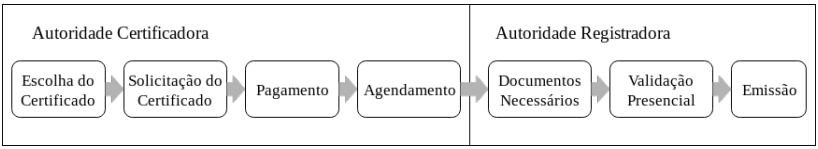

Figura 2 – Fluxo do processo de emissão de certificados digitais
Fonte: Gomes (2017)

2.6.2 Detecção de Irregularidades no processo de emissão

No dia 31 de março de 2016 foi divulgado um documento intitulado "Procedimentos para identificação do requerente e comunicação de irregularidades no processo de emissão de um certificado digital ICP-Brasil"que define atividades a serem executados no processo de Validação Presencial (ITI, 2017d). Essas atividades são descritas sucintamente a seguir:

- Primeiro deve ser feito um registro inicial, no qual o AGR confirma a identidade do indivíduo ou representante da organização, ou seja, que a pessoa que se apresenta como titular do certificado de pessoa física é realmente aquela cujos dados constam na documentação apresentada. No caso de pessoa jurídica, comprovar

que a pessoa física que se apresenta é a representante legal da organização.

- O AGR também confere se os dados da solicitação de certificado coincidem com os que estão nos documentos apresentados. Depois disso, é preciso efetuar a autenticação da identidade do solicitante. Entre os documentos aceitos:

  - Cédula de Identidade com foto (normalmente RG ou CNH), ou Passaporte, caso seja brasileiro;
  - Carteira Nacional de Estrangeiro (CNE), caso seja estrangeiro domiciliado no Brasil;
  - Passaporte, se estrangeiro não domiciliado no Brasil;
  - Comprovante de residência/domicílio, emitido há no máximo três meses da data da validação presencial;
  - Mais um documento oficial com fotografia, no caso de tipos de certificado A4 e S4;
  - Impressões digitais do requerente;
  - Fotografia da face do requerente.

- Os AGRs devem fazer uma análise minuciosa da cédula de identificação, principalmente do RG e CNH. As ACs deverão implementar alguma forma (consultas a bases oficiais, auxílio de peritos e/ou softwares) de consulta para validação de dados biográficos contidos no documento de identidade.

- Os resultados sem irregularidades da validação anterior deverão ser adicionados ao dossiê do titular do certificado. Caso a AR conclua pela validade do documento de identificação, deve prosseguir com o processo de emissão do certificado digital. Caso a AR conclua pela não validade do documento, deve comunicar a AC para que essa faça o comunicado de tentativa de fraude ao ITI.

- As ACs devem disponibilizar, para suas ARs vinculadas, uma interface para consulta a base de dados da Lista Negativa da AC, por meio do próprio sistema de emissão de certificados, com os mesmos requisitos de segurança e disponibilidade, em cada processo de emissão de um certificado digital ICP-Brasil. Lembrando que, caso ocorra qualquer indisponibilidade no banco de dados da Lista Negativa da AC, não deve ser emitido o certificado digital.

> Lista Negativa é um conjunto de informações derivadas dos comunicados de fraude, ou indícios de fraude, feitos pelas AC (ou pelo próprio ITI por meio de auditoria e fiscalização) da ICP-Brasil ao ITI, em que contém o modo de operação da ocorrência, as informações biográficas do documento apresentado e, se for o caso, das informações sobre a empresa, características fisiológicas do suposto fraudador, a imagem da face e do documento de identificação utilizado pelo suposto fraudador (ITI, 2017d).

- A interface para consulta à Lista Negativa deve disponibilizar para os AGR, no mínimo, algumas consultas especificadas:
  - Consulta aos dez maiores supostos fraudadores da ICP-Brasil;
  - Consulta aos comunicados de indícios ou fraudes dos últimos sete dias, considerando:
    - UF e cidade em que ocorreu o indício ou fraude;
    - Relato e data da ocorrência;
    - Modo como foi detectado o indício ou fraude;
    - Dados apresentados no documento de identificação da pessoa física;
    - Características físicas;
    - Informações da empresa;
    - Imagem de todo documento de identificação da ocorrência.
  - Pesquisas pelas características físicas notoriamente visíveis do requerente. Caso a pesquisa apresente muitos resultados, e não haja certeza sobre a inclusão de outras características físicas, os AGRs devem relacionar essa pesquisa a outros campos. Entre essas características estão:
    - Cor da pele;
    - Cor dos olhos;
    - Cor predominante do cabelo;
    - Deficiências físicas perceptíveis;
    - Idade aparente;
    - Sexo;

    - Sinais corporais perceptíveis;
    - Tipo de cabelo.
  - Pesquisas pelas informações biográficas fornecidas pelo requerente: nome, CPF, correio eletrônico, razão social e CNPJ. Caso não se obtenha qualquer resultado, deve ser realizada uma busca por fraudadores na região em que a AR está operando. Essa região pode, também, estender-se por UFs próximas ou mais específicas como a municípios próximos. Caso essa pesquisa apresente um resultado muito extenso, é recomendável que se adicione outros campos de características físicas do requerente.

- Se os resultados das pesquisas concluam pela ausência do requerente do certificado digital na Lista Negativa, os AGRs devem prosseguir com as validações e verificações.

- Na situação em que os resultados das consultas constatem que o requerente do certificado digital integra a Lista Negativa, com a imagem da face e/ou do documento de identificação coincidente com o apresentado pelo requerente, os AGRs devem realizar as validações e verificações elencadas, preferencialmente, comunicar à AC vinculada para que se faça uma análise detalhada do caso.

  - Caso a AR e/ou a AC concluam pela não emissão do certificado digital, a AC deve comunicar a tentativa de fraude ao ITI;
  - Caso a AR e/ou a AC concluam pela emissão do certificado digital, a AC deve solicitar o cancelamento de fraude, ou tentativa, na Lista Negativa, embasando detalhadamente os motivos de tal.

- Se os resultados das pesquisas à Lista Negativa tenham encontrado as informações biográficas do requerente e/ou da empresa, com a imagem da face e/ou do documento de identificação não coincidente com o apresentado pelo requerente, os AGRs, além de realizarem as validações e verificações elencadas, devem comunicar à AC vinculada para que se faça uma análise detalhada do caso.

  - Caso a AR e a AC concluam que o requerente se trata do titular de fato do documento de identificação e/ou das informações da empresa, deve prosseguir com o processo de emissão do certificado digital;

– Caso a AR e a AC concluam que se trata de outro suposto fraudador, utilizando as informações biográficas da pessoa e/ou da empresa já cadastradas no banco de dados da Lista Negativa, não se deve emitir o certificado digital e a AC deve comunicar a tentativa de fraude ao ITI.

A figura 3 representa o fluxo de atividades efetuadas pelo AGR para verificar se o requerente consta na Lista Negativa.

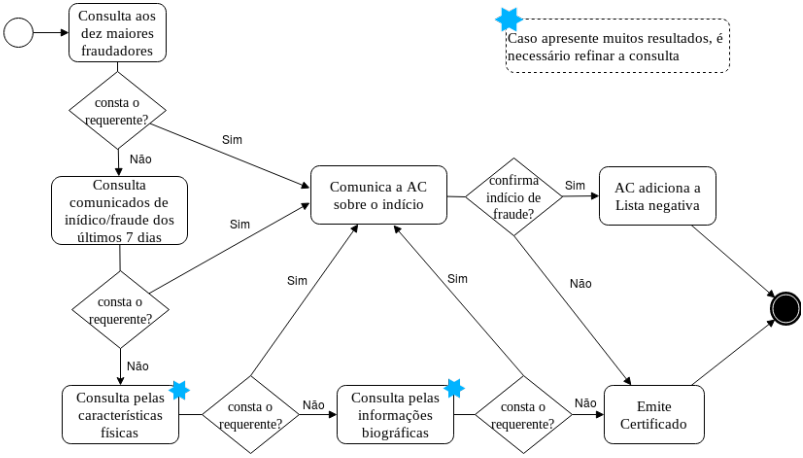

Figura 3 – Fluxo do processo de consulta à Lista Negativa

- Além da Lista Negativa, as ACs devem disponibilizar, para todas as suas ARs vinculadas, uma interface para verificação biométrica do requerente junto ao Sistema Biométrico da ICP-Brasil, para também ser utilizado no processo de emissão. Considerando o foco da proposta, não serão detalhadas as atividades relacionadas a verificação biométrica dos requerentes.

- Após encontrada e confirmada a irregularidade, a AC deve utilizar o Sistema de Comunicação de Fraude, preenchendo as informações necessárias e enviando-as ao ITI para alimentar o banco de dados da Lista Negativa. Neste formulário são encontrados os seguintes campos:

  – A AC e AR onde ocorreu a fraude ou tentativa - obrigatório;

- Nome do Informante: quem está cadastrando a fraude - opcional;
- CPF do Informante - opcional;
- UF onde ocorreu fraude ou indício - obrigatório;
- Município - obrigatório;
- Tipo de Ocorrência: se foi indício ou fraude - obrigatório;
- Número do certificado caso tipo de ocorrência for fraude - obrigatório;
- Ocorrência, breve relato do modo de operação do estelionatário - obrigatório;
- Data da ocorrência, ou seja, data do comunicado de fraude/indício - obrigatório;
- Diligência de investigação, como foi detectada a fraude (análise do documento) - opcional;
- Nome conforme aparece no documento apresentado - obrigatório;
- CPF conforme apresentado no documento - obrigatório;
- Data de nascimento conforme apresentado no documento - obrigatório;
- Correio eletrônico fornecido do suposto fraudador - opcional;
- Telefone fornecido - opcional;
- Documento de identidade, caso seja RG/Carteira militar fornecer número e data de expedição - obrigatório se for o caso;
- CNH, fornecer as seguintes informações: número, data de emissão, 1ª habilitação, UF expedição,data de validade, formulário, número de identidade - obrigatório, se for o caso;
- Passaporte, fornecer as seguintes informações: número, data de expedição, data de validade, país - obrigatório, se for o caso;
- Carteira de Trabalho e Previdência Social (CTPS), fornecer as seguintes informações: número, data de emissão, Programa de Integração Social/Programa de Formação do Patrimônio do Servidor Público (PIS/PASEP), UF - obrigatório, se for o caso;

  - Qualquer outro documento de natureza civil, que têm por força legal a presunção de identificação, fornecer as seguintes informações: número, data de emissão, nome, UF - obrigatório, se for o caso;
  - Características físicas perceptíveis:
    * Cor da pele (seleção: amarelo; branco; indígena; negro; pardo);
    * Cor dos olhos (seleção: claros; escuros);
    * Cor predominante do cabelo (seleção: branco; escuro; grisalho; loiro; ruivo);
    * Deficiências físicas perceptíveis (seleção: cadeirante; cego; manco; mudo; surdo);
    * Idade aparente (seleção: menor que 30 anos; entre 30 e 50 anos; mais de 50 anos);
    * Sexo (seleção: masculino; feminino);
    * Sinais corporais perceptíveis (seleção: falta de dedos nas mãos; mancha na pele; marcas como cicatrizes; tatuagem ou sinais em membros superiores; tatuagem ou sinais no rosto ou pescoço);
    * Tipo de cabelo (seleção: calvo; curto; longo; médio) - opcional;
  - Informações da empresa, CNPJ, razão social, endereço, telefone, CEP, CNAE, UF, Município - obrigatório, se for o caso;
  - Upload de imagem do documento de identificação e da face disposta em pé do suposto fraudador no comunicado - obrigatório;

- A AC emissora do certificado digital deve cuidar para que se notifique a fraude a autoridade policial competente mais próxima.

2.7 DADO, INFORMAÇÃO E CONHECIMENTO

Muitas decisões são tomadas por intuição de pessoas, sem levar em consideração o conhecimento incorporado nos dados. Dados são elementos brutos, sem significado (DAVENPORT; PRUSAK, 1998). É uma estrutura para registrar o evento, geralmente uma transação, e serve como matéria-prima para a informação. Geralmente armazenados em sistemas tecnológicos, são facilmente quantificados.

Informação são dados com significado, relevância e propósito. Geralmente os dados são processados e contextualizados (ANGELONI, 2003), adquirindo "forma"para auxiliar o entendimento de quem recebe esta. A informação tem objetivo de impactar no modo como o destinatário vê algo, interferindo em seu julgamento (DAVENPORT; PRUSAK, 1998).

Para explicar o que é o conhecimento, Angeloni (2003) define:

> O conhecimento pode então ser considerado como a informação processada pelos indivíduos. O valor agregado à informação depende dos conhecimentos anteriores desses indivíduos. Assim sendo, adquirimos conhecimento por meio do uso da informação nas nossas ações.

O maior desafio de sistemas que auxiliam na tomada de decisão é o de transformar dados em informação, e informação em conhecimento, minimizando as interferências dos indivíduos nesse processo (ANGELONI, 2003).

2.8 DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS

Tradução para *Knowledge Discovery in Databases* (KDD). Basicamente é o processo utilizado para identificar padrões novos, válidos, úteis e compreensíveis a partir de conjuntos de dados complexos e grandes. Este processo é iterativo, pois as vezes é necessário voltar a etapas anteriores, e interativo, pois não é um processo isolado, as escolhas certas dependem de cada passo e do domínio da aplicação (MAIMON; ROKACH, 2005).

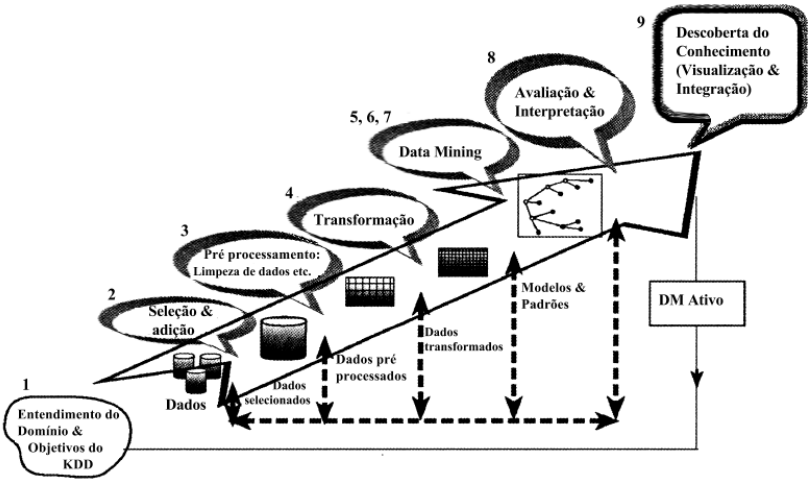

Figura 4 – Processo de KDD
Fonte: Maimon e Rokach (2005)

Como podemos observar na Figura 4, o processo começa com a definição dos objetivos do KDD, e termina com a implementação do conhecimento descoberto, podendo ser feitas alterações no domínio da aplicação estudada. Isso fecha o ciclo, e as consequências disso são medidas novamente através de um novo processo de KDD nos novos conjuntos de dados (MAIMON; ROKACH, 2005).

A seguir uma breve descrição das nove etapas do processo:

1. **Compreensão do domínio da aplicação e os objetivos**: Entender o que deve ser feito para, com base nisso, tomar as melhores decisões. É preciso definir os objetivos do usuário final e entender o ambiente em que o KDD ocorrerá.

2. **Seleção e criação do conjunto de dados**: Descobrir quais dados estão disponíveis, obter dados adicionais se possível, e definir quais atributos serão utilizados no processo. É uma etapa muito importante, pois é a base de evidências para a construção dos modelos de *Data Mining*. A falta de atributos importantes pode afetar o sucesso do estudo. Por outro lado, coletar, organizar e operar bancos de dados complexos é custoso.

3. **Pré-processamento e limpeza**: Essa etapa aumenta a confiabilidade dos dados, e também interfere na eficiência dos modelos

que serão aplicados posteriormente. São identificados e removidos atributos com muitos valores ausentes, ruídos e *outliers*.

4. **Transformação dos dados**: São utilizados métodos como discretização (alterar atributos para faixas de valores), normalização (ajustar as escalas de valores dos atributos para o mesmo intervalo), e outras transformações de tipos de atributos. Essa etapa serve para melhorar os dados antes da mineração.

5. **Escolha da tarefa de *Data Mining* apropriada**: Nessa etapa define-se qual tipo de *Data Mining* deve ser feito. Entre as principais tarefas estão classificação, regressão, agrupamento e associação. A escolha depende muito dos objetivos do KDD e do conjunto de dados disponível. Existem dois objetivos maiores em *Data Mining*: **descrição** ou **predição**. Tarefas preditivas normalmente utilizam **aprendizado supervisionado**, com dados já classificados, divididos em treinamento e teste. Enquanto tarefas descritivas utilizam **aprendizado não supervisionado**.

6. **Escolha do algoritmo de *Data Mining***: Definida a estratégia, agora é a etapa em que serão definidas as táticas. Isso inclui a escolha do método específico para encontrar padrões. Dentre esses algoritmos, estão redes neurais, árvores de decisões, cálculos de similaridade para agrupamentos. Cada algoritmo possui suas peculiaridades, e elas devem ser levadas em consideração.

7. **Implementação do algoritmo de *Data Mining***: Nessa etapa o algoritmo é executado até atingir um resultado satisfatório. Para isso são considerados os parâmetros do algoritmo, o número de instâncias, etc.

8. **Avaliação**: Etapa para avaliar e interpretar os padrões minerados, considerando os objetivos definidos na primeira etapa. Caso a avaliação não seja boa, pode-se retornar a etapas anteriores para melhorar os dados, trocar de algoritmo, etc.

9. **Utilização do conhecimento descoberto**: Nessa etapa é feita a incorporação do conhecimento útil descoberto. Assim é possível efetuar modificações no domínio da aplicação, para depois medir os efeitos que foram gerados a partir desse conhecimento.

Outro padrão de processo para descoberta de conhecimento é o *Cross Industry Standard Process for Data Mining* (CRISP-DM), que

difere em alguns pontos com relação ao KDD, como no número de etapas. A Figura 5, a seguir, mostra o fluxo deste processo:

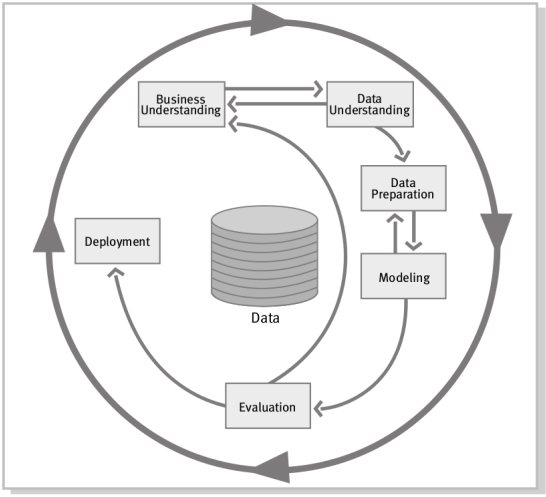

Figura 5 – Fases do modelo CRISP-DM
Fonte: Chapman et al. (2000)

A Tabela 1 mostra as equivalências das etapas dos dois processos.

Tabela 1 – Etapas do KDD e CRISP-DM

KDD	CRISP-DM
Compreensão do domínio	Entendimento do negócio
Seleção e criação do conjunto de dados	Entendimento dos dados
Pré processamento	
Transformação	Preparação dos dados
Data Mining	Modelo DM
Interpretação e avaliação	Avaliação
Utilização conhecimento	Desenvolvimento

Fonte: Azevedo (2008)

2.8.1 Mineração de Dados

Mineração de dados, mais popularmente conhecido no inglês *Data Mining*, é considerado o núcleo do KDD (MAIMON; ROKACH,

2005). É um processo essencial que inclui a execução de algoritmos complexos e inteligentes que buscam associações e padrões existentes que podem ser úteis de alguma forma (PEI; HAN; KAMBER, 2012).

Inicialmente é fundamental entender os diferentes propósitos e métodos de *Data Mining*, que Maimon e Rokach (2005) definem como "paradigmas". O objetivo da mineração de dados é comumente dividido em dois principais: **descoberta** (para encontrar novas regras e padrões de forma autônoma) e **verificação** (para validar a hipótese do usuário).

Dentro dos métodos de descoberta, Maimon e Rokach (2005) definem que existem dois distintos: **predição** e **descrição**. Métodos de descrição são utilizados para interpretação dos dados, com foco na compreensão da forma como os dados se relacionam. Já os métodos de predição geralmente constroem um modelo que é capaz de prever valores de um ou mais atributos de novos registros.

A seguir é possível visualizar as divisões e subdivisões dos paradigmas de *Data Mining* na Figura 6.

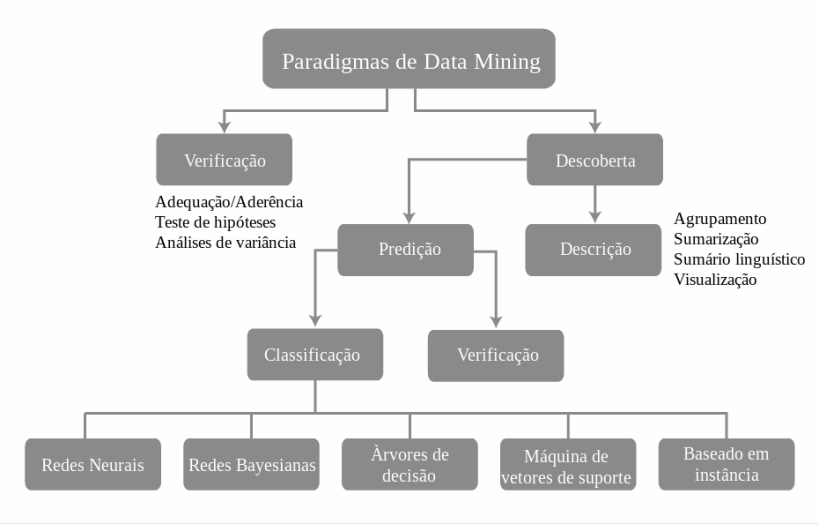

Figura 6 – Paradigmas de *Data Mining*
Fonte: Maimon e Rokach (2005)

A maioria das técnicas de descoberta são baseadas em aprendizagem induzida, ou seja, a inferência do conhecimento é feita com base na análise dos dados. Outra terminologia utilizada, no que refere-se aos métodos preditivos, é a aprendizagem supervisionada e a não-

supervisionada. Pei, Han e Kamber (2012) definem **aprendizagem supervisionada** como um método possuindo um conjunto de dados de treino, no qual as classes já foram definidas, para então ser gerado o modelo com a regra de classificação que poderá classificar novos registros; e **aprendizagem não-supervisionada** como um método que dispensa dados já classificados, que apenas observa os dados e reconhece padrões por si só, e o resultado disso são classes reconhecidas. Os principais modelos de aprendizagem supervisionada são classificação e regressão.

Pei, Han e Kamber (2012) explicam que as técnicas de classificação são divididas em dois tipos: **classificadores *eager* (ansiosos)** que, a partir do conjunto de treinamento, constroem o modelo de classificação e essa amostragem não é mais utilizada para novas classificações. Este tipo engloba técnicas como: árvore de decisão, redes neurais, redes bayesianas, máquinas de vetores de suporte e regras de decisão. O segundo tipo são os **classificadores *lazy* (preguiçosos)**, onde cada novo registro é comparado com todo o conjunto de treinamento e classificado com a classe do registro mais similar. Exemplos de classificadores *lazy* são *k-nearest-neighbor* e raciocínio baseado em casos.

Como o objetivo do trabalho é a predição da fraude com os dados do requerente do certificado no processo de emissão, é fundamental compreender as técnicas utilizadas para classificação. Essas técnicas estão descritas nos subtópicos a seguir.

2.8.1.1 Árvores de decisão

Segundo Pei, Han e Kamber (2012), uma árvore de decisão é uma estrutura de árvore semelhante a um fluxograma. Cada nodo representa um atributo, logo cada nodo interno (não-folha) denota um teste em um atributo, cada aresta representa um resultado do teste e cada nodo folha (ou nodo terminal) representa um rótulo de classe.

Pode-se observar na Figura 7 um exemplo de árvore de decisão, que prevê se um cliente de determinada loja comprará um computador:

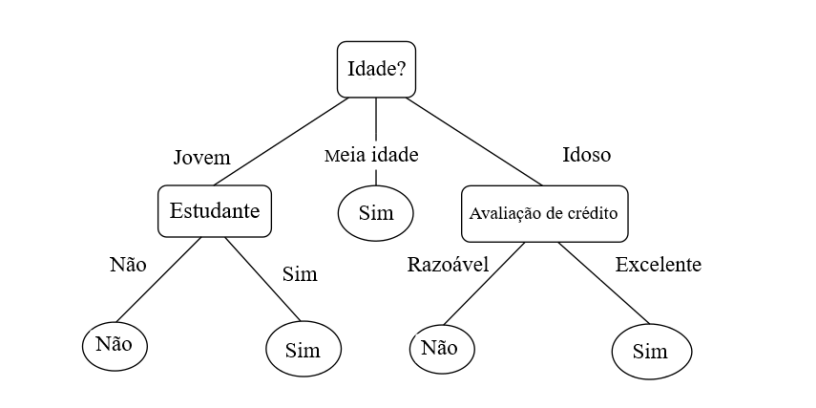

Figura 7 – Exemplo de árvore de decisão
Fonte: Pei, Han e Kamber (2012)

Para efetuar a classificação, uma tupla sem classificação tem os seus valores de atributo testados considerando a árvore de decisão. Um caminho é traçado da raiz para um nodo folha, o que resulta na previsão da classe para essa tupla. Uma das vantagens dessa técnica é que sua representação do conhecimento em forma de árvore é intuitiva e fácil de assimilar, e também pode ser convertida em regras.

Durante a construção da árvore, algumas medidas de seleção de atributos são usadas para escolher o atributo que melhor particiona as instâncias em classes distintas. Entre as medidas mais utilizadas estão o cálculo de ganho de informação e o índice de Gini.

2.8.1.2 Redes Bayesianas

Os classificadores que utilizam redes bayesianas são basicamente classificadores estatísticos. Pei, Han e Kamber (2012) mencionam que este baseia-se no Teorema de Bayes, o qual considera a probabilidade de uma hipótese H dada a observação de uma evidência E e a probabilidade da evidência dada pela hipótese, como podemos observar na fórmula 2.1:

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \tag{2.1}$$

A classificação Naïve Bayes faz a análise de acordo com a probabilidade condicional do teorema de Bayes. Considerando as classes

"sim"e "não", o cálculo das duas hipóteses seria efetuado sobre uma determinada tupla, com o objetivo de determinar a qual das classes a tupla possui maior probabilidade de pertencer. A evidência pode ser separada em partes independentes, pois assume-se que o efeito de um valor de atributo em uma determinada classe é independente dos valores dos outros atributos, como pode ser visto na fórmula 2.2:

$$P(H|E) = \frac{P(E_1|H) \times P(E_2|H)...P(E_n|H) \times P(H)}{P(E_1) \times P(E_2)...P(E_n)} \tag{2.2}$$

Pei, Han e Kamber (2012) citam que como vantagem desses classificadores o fato de apresentarem alta velocidade de resposta e boa precisão com bases de dados grandes. Entretanto, não apresenta bons resultados em problemas de maior complexidade.

2.8.1.3 Redes Neurais

Pei, Han e Kamber (2012) indicam que essa técnica originalmente foi incentivada por psicólogos e neurologistas, que queriam desenvolver e testar análises computacionais de neurônios. Luger (2013) afirma que a base de redes neurais é o neurônio artificial, representado na Figura 8, que é composto por:

- **Sinais de entrada x_i**: dados do ambiente ou da ativação de outros neurônios;
- **Conjunto de pesos w_i**: descrevem as forças de conexão;
- **Nível de ativação $\sum w_i x_i$**: é a soma ponderada das entradas;
- **Função limiar**: calcula o estado final ou de saída do neurônio, quando o nível de ativação está acima ou abaixo do limiar, reproduzindo o estado ligado/desligado de um neurônio real.

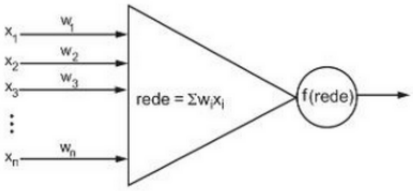

Figura 8 – Exemplo de neurônio artificial
Fonte: Luger (2013)

Pei, Han e Kamber (2012) afirmam que uma rede neural é um conjunto de unidades de entrada/saída conectadas nas quais cada conexão tem um peso associado. Durante a fase de aprendizagem, a rede aprende ajustando os pesos para poder prever o rótulo de classe correto das tuplas de entrada. Redes neurais exigem uma série de parâmetros, como a topologia de rede, que normalmente são melhor determinados de forma empírica, através de experiências e considerando o contexto da aplicação.

Algumas das desvantagens das redes neurais são a demora para efetuar o treinamento, e também devido a sua fraca interpretabilidade. As vantagens das redes neurais, no entanto, incluem a sua alta tolerância a dados com ruídos. Estas técnicas foram bem-sucedidas ao serem aplicadas no reconhecimento de caracteres manuscritos, patologia e medicina de laboratório, e treinamento de um computador para pronunciar.

2.8.1.4 Máquina de vetores de Suporte

Máquina de vetores de suporte (do inglês "*Support Vector Machines*", SVM) é um algoritmo de aprendizado de máquina supervisionado. Pei, Han e Kamber (2012) explicam o funcionamento desta técnica, que inicialmente coloca cada tupla/objeto como um ponto em um espaço n-dimensional, onde n são os atributos deste objeto.

Esse algoritmo pode ser utilizado para a situação em que os dados são linearmente separáveis, ou seja, uma linha reta é capaz de separar os objetos nas respectivas classes. Como existe um número infinito de linhas que conseguem separar os pontos, é necessário encontrar a que possui a menor taxa de erro de classificação. Na Figura 9 a seguir pode-se observar que duas linhas possuem margens diferentes.

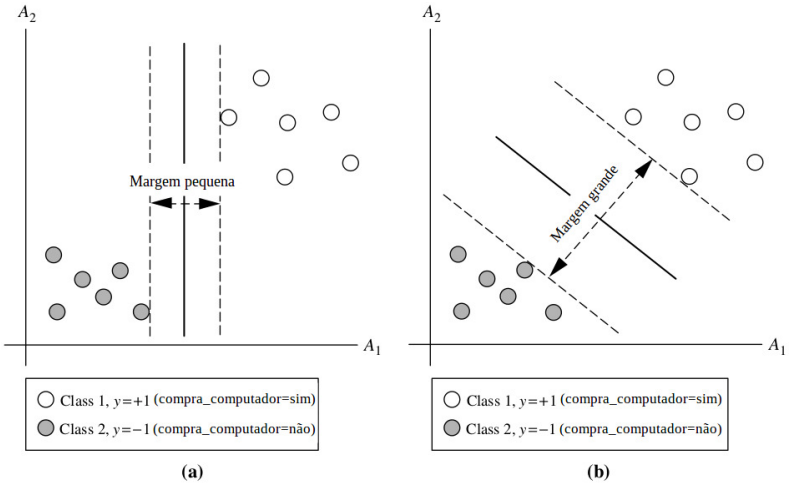

Figura 9 – Exemplo de margem SVM
Fonte: Pei, Han e Kamber (2012)

Hiperplano é o termo utilizado para generalizar o espaço com n dimensões, e este é o objetivo do SVM: encontrar o melhor hiperplano e os vetores de suporte. Todas as tuplas de treinamento que caem em hiperplanos (ou seja, os "lados"que definem a margem) são chamados de vetores de suporte.

Durante a aprendizagem ou treinamento o SVM procura o hiperplano com a maior margem, ou seja, o hiperplano marginal máximo (do inglês "*Maximal Margin Hyperplane*", MMH) que fornece a maior separação entre as classes, utilizando a otimização quadrática restrita, que é uma questão matemática e não cabe ao escopo deste trabalho.

Para a situação em que os dados não são linearmente separáveis, a abordagem de SVMs lineares pode ser estendida. Uma função *kernel* é utilizada com o objetivo de mapear os dados originais em um espaço dimensional maior.

Embora o tempo necessário para treinamento seja grande, SVMs são altamente precisos. Foram aplicados em várias áreas, incluindo reconhecimento de dígitos manuscritos, de objetos, entre outros.

2.8.1.5 Classificadores baseados em instâncias

Como descrito anteriormente, Pei, Han e Kamber (2012) definem classificadores baseados em instância como *lazy*, pois aprendem a partir das instâncias "vizinhas". Dois classificadores se encaixam nessa definição: *K-nearest-neighbor* (K-NN) e raciocínio baseado em casos.

Um classificador K-NN possui um conjunto de tuplas de treinamento, que são descritas por n atributos, e cada tupla é representada como um ponto em um espaço n-dimensional. Esse classificador compara uma nova tupla com todas as tuplas de treinamento, utilizando alguma medida de similaridade, geralmente a distância euclidiana. Assim, é possível definir os k "vizinhos mais próximos", e a classe da maioria dos vizinhos é atribuída a essa nova tupla.

Um classificador baseado em casos, diferente do K-NN em que as tuplas representam pontos em um espaço, trata as tuplas como casos e as armazena como descrições simbólicas complexas. Ao receber uma nova tupla para classificação, o classificador baseado em casos verifica se há um caso idêntico no conjunto de treinamento. Se sim, é retornada a classificação deste caso. Se nenhum caso idêntico for encontrado, o classificador buscará casos de treinamento com componentes semelhantes. Conceitualmente, esses casos de treinamento podem ser considerados como "vizinhos"do novo caso.

2.8.1.6 Métricas para avaliar o desempenho do classificador

Pei, Han e Kamber (2012) descrevem algumas métricas para medir o quão bom o classificador é ao predizer as classes das novas tuplas. Antes de descrever as métricas, é preciso entender algumas terminologias. Supondo que se tem um conjunto de dados com classes binárias (Ex: "sim"e "não"), consideram-se os seguintes termos:

- **Verdadeiro positivo (VP):** Número de tuplas positivas que foram classificadas corretamente como positivas.
- **Verdadeiro Negativo (VN):** Número de tuplas negativas que foram classificadas corretamente como negativas.
- **Falso Positivo (FP):** Número de tuplas negativas que foram classificadas incorretamente como positivas.
- **Falso Negativo (FN):** Número de tuplas positivas que foram classificadas incorretamente como negativas.

Estes termos são, então, resumidos na **Matriz de Confusão**. A Matriz de Confusão é uma ferramenta muito útil quando se deseja analisar a eficácia e a eficiência de um classificador. Enquanto VP e VN indicam as entradas corretamente classificadas, FP e FN, as incorretamente classificadas. Podemos observar esta matriz de confusão na Tabela 2 a seguir.

Tabela 2 – Exemplo de Matriz de Confusão considerando classes *sim* e *não*

		Classe predita		
		sim	*não*	Total
Classe real	*sim*	*VP*	*VN*	*P*
	não	*FP*	*FN*	*N*
	Total	*P'*	*N'*	*P + N*

Fonte: Pei, Han e Kamber (2012)

Explicados os indicadores, é possível compreender as métricas descritas por Pei, Han e Kamber (2012). Começando com **acurácia**, que é a porcentagem de entradas classificadas corretamente em relação ao total de entradas do conjunto e dá-se pela seguinte fórmula:

$$\frac{TP + TN}{P + N} \tag{2.3}$$

Outra métrica interessante é a **taxa de erros**, que é a porcentagem de entradas classificadas incorretamente em relação ao total de entradas do conjunto e dá-se pela seguinte fórmula:

$$\frac{FP + FN}{P + N} \tag{2.4}$$

Outras métricas amplamente utilizadas em classificação são a **precisão** e o ***recall***. Pei, Han e Kamber (2012) os descrevem como a exatidão (porcentagem de entradas classificadas como *sim* corretamente) e completude (porcentagem de entradas de classe *sim* classificadas corretamente) e são calculados pelas fórmulas 2.5 e 2.6, respectivamente.

$$\frac{VP}{VP + FP} \tag{2.5}$$

$$\frac{VP}{VP + FN} \tag{2.6}$$

3 TRABALHOS CORRELATOS

3.1 ESTUDO ABRANGENTE DE DETECÇÃO DE FRAUDE BASEADA EM MINERAÇÃO DE DADOS

Este artigo escrito por Phua et al. (2010) categoriza e compara as principais técnicas publicadas nos últimos 10 anos sobre detecção automática de fraude, assim como seus contextos. Também são elencados os tipos de fraudadores e as áreas da indústria que são atacadas.

O fraudador tem interesse lucrativo para querer afetar o negócio. A empresa está suscetível a fraudes externas ou internas, com a corrupção de seus funcionários. Além de auditorias internas, a mineração de dados pode ser utilizada como ferramenta analítica. O autor categoriza os fraudadores na seguinte hierarquia: este pode ser da gerência, subordinado ou externo.

Para Phua et al. (2010), o fraudador externo ainda pode ser categorizado como um delinquente médio (comete fraudes ocasionalmente e não possui um comportamento muito específico), criminoso ou de alguma quadrilha ou crime organizado. Ele pode cometer fraudes na forma de potencial cliente, ou fornecedor. Os fraudadores que se arriscam mais são os criminosos individuais ou de quadrilhas, porque cometem repetidamente as fraudes e tem o seu *modus operandi* desenvolvido ao longo do tempo para combater os sistemas de detecção.

O autor afirma, ainda, que fraudes "internas"e de seguro estão mais propensas a serem cometidas por delinquentes médios, enquanto nas áreas de telecomunicação e transação de crédito, por fraudadores profissionais. Nas áreas da indústria que sofrem fraudes, as mais afetadas são: telecomunicação, transação de crédito, gerenciamento e automobilística. A área de transações de crédito é a que mais recebe atenção dos pesquisadores na questão de detecção de fraude.

O propósito principal de sistemas de detecção de fraude é identificar tendências de aplicações/transações suspeitas. Fraudadores solicitam montantes de seguro usando informações falsificadas, ou solicitam produtos e serviços de crédito usando informações de identidades inexistentes ou de outra pessoa.

De acordo com Little et al. (2002), os atributos específicos para detectar cada tipo de fraude são geralmente os mesmos. Os dados típicos para detectar fraudes em gerenciamento são: índices financeiros, contas a receber, provisão de dívidas de cobrança duvidosa e número de vendas líquidas. Já para seguro doméstico, são utilizados dados sobre

o comportamento do cliente (montante do sinistro atual, tempo como cliente, reclamações passadas) e situação financeira (rendimento anual, saldo bancário médio).

Atributos específicos em dados de transações de crédito muitas vezes não são revelados, mas devem incluir data e hora, transação atual (valor, localização geográfica, código da indústria mercantil e código de validade), histórico transacional, histórico de pagamentos e outras informações de conta) (CHAN et al., 1999; GHOSH; REILLY, 1994).

Não são publicados conjuntos de dados para estudar a detecção de fraudes desses contextos. Obter dados reais de empresas para fins de pesquisa é uma tarefa extremamente difícil, devido a razões legais e o mercado competitivo. Para contornar esses problemas de disponibilidade de dados e trabalhar em um determinado tipo de fraude, uma alternativa é criar dados sintéticos que correspondam de perto aos dados reais.

Barse, Kvarnstrom e Jonsson (2003) justifica que dados sintéticos podem treinar e adaptar um sistema sem dados sobre fraudes conhecidas, variações de fraudes e novas fraudes podem ser criadas artificialmente, além de permitir comparar sistemas diferentes. Além disso, foram definidas qualidades importantes para dados simulados e há uma proposta de metodologia para geração de dados sintéticos em etapas.

Na questão de medida da performance do sistema de detecção de fraude, a maioria dos departamentos de fraude atribuem valor monetário para as previsões com objetivo de maximizar a economia de custos versus lucro de acordo com suas políticas. Cahill et al. (2002) sugere dar uma pontuação para uma instância, determinando a semelhança da mesma com os exemplos de fraude conhecidos, dividindo pela sua dissimilaridade a exemplos legais conhecidos (instância legítima).

A maioria dos estudos de detecção de fraude que utilizam algoritmos supervisionados abandonaram medidas como acurácia e precisão. Na detecção de fraude, os custos de erros de classificação (falsos positivos e falsos negativos) são desiguais, incertos, podem variar de exemplo para exemplo e podem mudar ao longo do tempo. "Na detecção de fraude, um falso negativo é geralmente mais caro do que um falso positivo"(PHUA et al., 2010).

Estudos recentes sobre a fraude transacional de cartões de crédito (CHEN et al., 2004) e de telecomunicações (KIM; ONG; OVERILL, 2003) ainda buscam apenas maximizar a precisão. Alguns usam a análise de ROC (*Receiver Operating Characteristic Curve*), que é a taxa positiva verdadeira versus taxa de falso positivo. Viaene, Derrig e De-

dene (2004), Foster e Stine (2004) procuram minimizar a pontuação de Brier (erro quadrático médio das previsões), enquanto Caruana e Niculescu-Mizil (2004) argumentam que a maneira mais eficaz de avaliar algoritmos supervisionados é usar uma métrica a partir de métricas de limiar, ordenação e probabilidade; e justificam o uso da média do erro quadrático médio, acurácia e AUC (*Area Under the Curve*). Já Fawcett e Provost (1999) recomendam características operacionais de monitoramento de atividades (AMOC).

Para abordagens semi-supervisionadas como a detecção de anomalia, Lee e Xiang (2001) propõem entropia, entropia condicional, entropia condicional relativa, ganho de informação e custo de informação. Para algoritmos não supervisionados Yamanishi et al. (2004) empregaram a pontuação de Hellinger e pontuações logarítmicas para encontrar *outliers* para dados de seguros; e Burge e Shawe-Taylor (2001) utilizaram a pontuação de Hellinger para determinar a diferença entre os perfis de curto e longo prazos da conta de telecomunicações. Além destes, Bolton, Hand et al. (2001) recomendam o uso da estatística-t como uma pontuação para computar a distância normalizada de uma determinada conta bancária com o ponto central de um agrupamento de transações similares, além de ser usado para detectar mudanças radicais de gastos entre contas.

Outras considerações importantes incluem a rapidez com que as fraudes podem ser detectadas (tempo de detecção e tempo de alarme), quantos tipos de fraude detectados, se a detecção foi feita em tempo real ou em lote (GHOSH; REILLY, 1994).

A maioria dos sistemas de detecção de fraude separa as ocorrências de fraudes em uma "lista negra" para comparar com instâncias novas. Uma ideia interessante proposta por Fawcett (2003) é entender a natureza temporal da fraude, a estratégia dos fraudadores, nos atributos das instâncias classificadas como fraude. Algumas questões a observar (FAWCETT, 2003):

- O volume de fraudes e classes legais crescem independentemente uns dos outros; Portanto, as distribuições de classe (proporção de exemplos ilegítimos para exemplos legítimos) mudarão ao longo do tempo;
- Vários estilos de fraude podem acontecer no mesmo período de tempo. Cada estilo pode ter uma característica temporal regular, ocasional, sazonal ou temporária;
- Depois de descobrir o *modus operandi* de fraudadores profissionais, esses mesmos fraudadores vão adaptar seus estilos de fraude,

até que os sistemas de detecção comecem a gerar falsos negativos novamente.

O artigo analisou quatro principais métodos comumente usados para detecção de fraudes, e suas correspondentes técnicas e algoritmos.

- **Abordagens supervisionadas em dados rotulados:** Os dados de treinamento rotulados podem ser processados por algoritmos supervisionados. Redes Neurais são populares nessa abordagem, como SVM e *Fuzzy*. Também é possível utilizar redes bayesianas, árvores de decisão, e regras de associação, entre outros algoritmos citados no artigo.

- **Abordagens híbridas com dados rotulados:** Uma sugestão melhor é empregar híbridos como múltiplos algoritmos supervisionados ou algoritmos supervisionados e não-supervisionados para produzir pontuações de suspeita, regras e/ou anomalias visuais nos dados de avaliação. Algoritmos supervisionados populares, tais como redes neurais, redes bayesianas e árvores de decisão foram combinados ou aplicados de forma sequencial para melhorar os resultados.

- **Abordagens semi-supervisionadas com apenas dados legítimos (não-fraudes):** Uma nova transação deve ser comparada com todas as não-fraudes para detectar anomalias significativas em relação ao comportamento normal.

  Existem algumas críticas sobre o uso de dados rotulados:

  - Em um ambiente operacional orientado a eventos, a eficiência do processamento é crítica;
  - O tempo necessário para marcar exemplos como fraudulentos será a mesma quantidade de tempo que os novos tipos de fraude passarão despercebidos;
  - Os rótulos de classe dos dados de treinamento podem estar incorretos;
  - Dados rotulados podem ser bastante caros e difíceis de obter;
  - Em alguns casos, os funcionários têm de rotular manualmente cada exemplo e isso tem o potencial de violar a privacidade, especialmente se os dados contiverem informações pessoais e de identidade.

- **Abordagens não-supervisionadas com dados sem rótulo:** Alguns autores recomendam o uso de dados não rotulados porque o fraudador tentará fazer com que a fraude e as classes legais sejam difíceis de distinguir. Combine dados de treinamento (os rótulos de classe não são necessários aqui) com dados de avaliação. Estes devem ser processados por algoritmos simples ou múltiplos não-supervisionados para gerar resultados de suspeita, regras e/ou anomalias visuais em dados de avaliação.

3.2 SISTEMA BASEADO EM *DATA MINING* PARA DETECÇÃO DE FRAUDES DE CARTÃO DE CRÉDITO EM *E-COMMERCE*

Carneiro, Figueira e Costa (2017) explicam neste artigo como a fraude é um grande problema para os comerciantes, especialmente no setor online. Segundo Bhatla, Prabhu e Dua (2003), fraude de cartão de crédito é a ação de um indivíduo que usa do cartão sem o consentimento do proprietário deste e sem intenção de reembolsar a compra feita. Os comerciantes são responsáveis por pagar a conta quando um fraudador rouba bens ou serviços em uma transação online dessas. Um estorno ocorre quando um consumidor afirma que não recebeu os produtos ou serviços solicitados, ou que o pedido foi solicitado por um fraudador.

Várias tecnologias têm sido usadas para evitar a ocorrência de fraudes, como o Sistema de Verificação de Endereços (AVS), a verificação de Chip e Pin e o Código de Verificação de Cartões (CVV). No entanto, mesmo estes sistemas avançados são suscetíveis a falhas. O desenvolvimento de métodos de detecção de fraudes é, portanto, de crucial importância.

Este é um problema desafiador porque os fraudadores fazem seus melhores esforços para que seu comportamento pareça legítimo. Outra dificuldade é que o número de registros legítimos é muito maior do que o número de casos fraudulentos (conjuntos desequilibrados exigem precauções extras). A chave para a precisão na detecção de fraude reside no desenvolvimento de sistemas dinâmicos que podem se adaptar a novos padrões de fraude.

Problemas como esse estão sendo abordados em diferentes contextos com diferentes abordagens:

- Verificar a identidade do cliente utilizando dados publicamente disponíveis;

- Compartilhamento de dados, que consiste em adquirir informa-

ções sobre o cliente de um provedor externo ou monitoramento de redes sociais;

- Serviços de tecnologia, tais como sistemas de biometria;
- Técnicas de mineração de dados que levam em conta as transações passadas para estimar a probabilidade de uma nova transação ser fraudulenta.

Uma diferença importante para o setor bancário é que um único varejista online tem informações muito limitadas sobre o cliente com o qual está fazendo negócios. Este artigo relata o desenvolvimento e a implementação de um sistema de detecção de fraude para um comerciante do ramo online.

Para o desenvolvimento, os autores utilizaram a metodologia CRISP-DM (CHAPMAN et al., 2000). Inicialmente, foram compreendidos os principais conceitos de negócios e objetivos envolvidos. A Segunda etapa envolveu a construção do conjunto de dados a partir do histórico de pedidos. Após isso, foi feita uma análise estatística exploratória, na qual identificaram possíveis padrões de fraude observando distribuições variáveis individuais.

Por um lado, a empresa tem de minimizar o nível de fraude, maximizando a detecção de transações fraudulentas, evitando assim, estornos. Por outro lado, deve fornecer altas taxas de aceitação de pagamentos para converter o maior número possível de vendas e minimizar o número de insultos a clientes (transações legítimas recusadas).

Indicadores-chave de desempenho relacionados à detecção de fraudes:

- O nível de automação: porcentagem de pedidos processados automaticamente;
- Nível de cobrança: porcentagem de ordens que originam um estorno;
- Taxa de pagamentos recusados: percentagem de pagamentos recusados;
- Velocidade de processamento: tempo necessário para aprovar ou rejeitar um pagamento de um pedido.

A abordagem proposta consiste na construção de um sistema de pontuação de risco baseado em métodos de aprendizado de máquina que estimará uma pontuação de suspeita de fraude para cada pedido.

A pontuação de suspeita estimado pelo modelo deve ser um número entre 0 e 1. O sistema de pontuação de risco também avaliaria se a pontuação cai abaixo de um certo limite (por exemplo, inferior a 30%), onde o pedido seria automaticamente aprovado. Ordens com maior pontuação seriam revisadas manualmente pela equipe de processamento de pedidos, que também poderia contar com a pontuação de suspeita para uma melhor avaliação.

As técnicas de mineração de dados dividem-se em duas abordagens principais: métodos supervisionados e não-supervisionados. Ambas as abordagens são baseadas no treinamento de um algoritmo com observações do passado. Métodos supervisionados exigem que cada uma dessas observações usadas para a aprendizagem tenha um rótulo indicando a qual classe ela pertence.

No contexto da detecção de fraude, isto significa que para cada observação sabemos se pertence à classe *fraudulenta* ou à classe *legítima*. Muitas vezes não sabemos a que classe pertence uma observação. Tais ocorrências favorecem o uso de métodos não-supervisionados, que não exigem que os dados sejam rotulados. Esses métodos procuram ocorrências extremas de dados ou *outliers*. A fim de obter o melhor de dois mundos, algumas soluções combinam técnicas supervisionadas e não-supervisionadas.

Neste estudo, foram escolhidos métodos de aprendizagem supervisionada para o problema de classificação, porque é comum que aplicativos de detecção de fraude tenham dados rotulados. Foram escolhidos para teste três modelos diferentes. Regressão logística devido à sua popularidade, e *Random Forests* e SVM, que foram usados em uma variedade de aplicações que apresentam desempenho superior. SVM mostrara-se bem nos problemas de classificação e *Random Forests* são muito promissores para a detecção de fraude devido à facilidade de aplicação e à computação eficiente.

Na etapa de compreensão e preparação dos dados, a manipulação dos dados foi feita utilizando-se de recursos e módulos pertencentes à linguagem de programação *Python*.

- Variáveis categóricas foram transformadas em numéricas;

- Uma decisão tomada foi: agrupar os países por risco de fraude, calculado pela proporção de pedidos fraudulentos em relação ao número total de pedidos nesse país;

- Foram criadas novas variáveis através da abstração e combinação de variáveis. Para isso, foram considerados os seguintes pares de

variáveis: país de cobrança e país de envio; país de cobrança e país do cartão; país de envio e país do cartão.

Como medidas de desempenho, os autores utilizaram a Matriz de confusão, as Análises ROC e AUC e a Análise das curvas de *Recall* de Precisão (PR). Como conjunto de treino e teste, foi utilizada a técnica de validação cruzada (*Cross-Validation*) com 10 grupos (*folders*).

A implantação do modelo selecionado envolve problemas adicionais, como a forma como o modelo será atualizado e a integração com os sistemas e serviços existentes do varejista. Para manter a relevância do classificador, é importante atualizar os dados utilizados para treinamento.

Para isso, é necessário executar dois procedimentos principais:

- Treinar o algoritmo: Treinar o algoritmo acontece uma vez por semana. Depois que BD foi atualizado, o *script* de treinamento irá baixar os registros. Como demora várias semanas antes do envio de um estorno, o *script* de treinamento usará apenas dados de pedidos de até quatro meses antes. Como não há necessidade de um conjunto de testes, todos os dados baixados são usados no treinamento. O resultado do processo é um classificador que gerará a pontuação de suspeita e o conjunto de parâmetros para a transformação de dados (como o risco de fraude nos países, etc);

- Classificar ordens: O serviço de classificação é executado continuamente. Quando um novo pedido é feito, este é submetido para avaliação através da API do serviço de classificação.

    1. Em primeiro lugar, o serviço analisa as informações na solicitação para mapear cada campo para a respectiva variável;
    2. Em seguida, ele aplica as funções de transformação aos dados, de modo que ele gere exatamente os mesmos recursos que no treinamento. Os parâmetros pré-processados salvos do treinamento, como o mapeamento do nível de risco dos países são usados nesta etapa.
    3. Finalmente, com todos os recursos no formato correto, o classificador é chamado para estimar a pontuação de suspeita de fraude para o pedido. Este resultado é retornado como a resposta da solicitação de classificação.

De acordo com Carneiro, Figueira e Costa (2017), todos os três algoritmos pré-selecionados foram bons, sendo *Random Forests* o que

obteve melhor desempenho. Este algoritmo parece ser muito adequado para ser usado para detecção de fraude, não só por causa de seu bom desempenho, mas também devido à facilidade de implementação e ao rápido tempo de computação, mesmo com grandes conjuntos de dados.

Carneiro, Figueira e Costa (2017) consideraram os resultados do sistema de classificação como satisfatórios e concluíram, também, que o uso de um conjunto de dados menor não contribuiu tanto quanto o esperado em termos de performance em relação ao uso do conjunto de dados completo e desbalanceado.

4 PROPOSTA DE DETECÇÃO DE FRAUDES

A proposta consiste na definição de um modelo para detecção de fraudes no processo de emissão de certificados na ICP-Brasil, que mostre-se mais eficiente que o modelo atual e que não dependa da intuição de indivíduos humanos. Para a arquitetura desta proposta e para o modelo de *Data Mining* de classificação da fraude, são considerados os seguintes tipos de fraudadores:

- Fraudador externo: a pessoa que deseja emitir o certificado;
  - A pessoa apresenta o documento de identificação original, mas não é a proprietária do mesmo;
  - A pessoa apresenta o documento de identificação falso.
- Fraudador interno: agente de registro que foi corrompido.
  - O AGR, deliberadamente, não reporta a fraude.

4.1 MODELO DE *DATA MINING*

O modelo de *Data Mining* foi definido com base no processo de KDD. A primeira etapa de compreensão do domínio da aplicação e os objetivos contemplou-se na revisão bibliográfica deste trabalho, que permitiu o entendimento dos cenários descritos acima. Com isso, identificou-se que o objetivo do *Data Mining* é predizer se uma nova solicitação de emissão de certificado digital trata-se de uma fraude ou não. Aqui são elencadas algumas formas de identificar as tentativas de fraudes:

- Em ambos os casos de fraudador externo, é possível verificar se:
  - A pessoa na foto do documento de identificação é diferente da pessoa na foto capturada durante a validação presencial;
  - Assinatura do documento de identificação é diferente da assinatura colhida na validação presencial;
  - A digital presente no documento de identificação é diferente da digital colhida na validação presencial.
- Para a verificação de uma fraude no documento, também pode-se verificar se a data de expedição do documento foi em um fim de semana ou feriado.

A segunda etapa do KDD engloba a seleção e criação do conjunto de dados. Para isso foi feito um levantamento de todos os dados disponíveis, tanto do requerente do certificado quanto das Autoridades envolvidas, para posterior seleção dos atributos:

- Nome do requerente;
- Imagem do requerente presente no documento de identificação;
- Foto do requerente capturada na validação presencial;
- Assinatura do requerente presente no documento de identificação;
- Assinatura do requerente colhida na validação presencial;
- Digital do requerente presente no documento de identificação;
- Digital do requerente colhida na validação presencial;
- Data de nascimento;
- CPF do requerente;
- Data de emissão do RG;
- Data de emissão da CNH;
- Data de emissão da CNE;
- Data de emissão do Passaporte Brasileiro;
- Data de emissão do CNPJ (caso haja);
- Data de emissão da Carteira do Conselho Profissional (caso haja);
- Data de emissão da Certidão de Nascimento (caso haja);
- Data de emissão da Carteira de Trabalho (caso haja);
- Política do certificado (A1, A2, A3, etc.);
- Cidade de residência do requerente;
- UF de residência do requerente;
- Razão social da AC;
- Razão social da AR.

A terceira etapa do KDD refere-se ao pré-processamento e limpeza, onde atributos irrelevantes são retirados. Neste caso, os atributos "Nome do requerente"e "CPF do requerente"foram descartados. Um bom uso para o CPF do requerente seria a validação em conjunto com bases de dados externas confiáveis para comparação de informações referentes à identidade. Esta validação foi descartada por ser considerada fora do escopo do trabalho.

Na quarta etapa do KDD é feita a transformação dos atributos para melhorá-los: foram definidas medidas de similaridade entre as imagens do requerente, similaridade entre assinatura digitalizada e a assinatura colhida, assim como a similaridade das digitais; a data de nascimento é convertida em um número que será utilizado para a criação de intervalos relevantes pelo algoritmo de *Data Mining* (número de milissegundos de diferença da data em relação ao dia 1º de janeiro de 1970); as datas de emissões dos documentos foram convertidas em valores lógicos indicando se são dias úteis ou não; foram gerados índices de ocorrências de fraudes em relação ao total de ocorrências para a cidade e UF de residência do requerente, bem como para a razão social da AC e da AR. Logo, o modelo de *Data Mining* utilizou os seguintes atributos:

- Similaridade da foto do documento de identificação com a foto capturada;
- Similaridade da assinatura do documento de identificação com a assinatura colhida;
- Similaridade da digital do documento de identificação com a digital colhida;
- Data de nascimento numérica;
- Data de emissão do RG não é um dia útil;
- Data de emissão do CNH não é um dia útil;
- Data de emissão CNE não é um dia útil;
- Data de emissão Passaporte Brasileiro não é um dia útil;
- Data de emissão do CNPJ não é um dia útil;
- Data de emissão da Carteira do Conselho Profissional não é um dia útil;

- Data de emissão da Certidão de Nascimento não é um dia útil;
- Data de emissão da Carteira de Trabalho não é um dia útil;
- Índice de ocorrência de fraudes na cidade de residência;
- Índice de ocorrência de fraudes no estado de residência;
- Índice de ocorrência de fraudes pela razão social da AC;
- Índice de ocorrência de fraudes pela razão social da AR.

É possível observar que foram consideradas as datas de todos os documentos separadamente. Isso foi feito porque múltiplos documentos podem ser entregues para a validação.

Durante a quinta etapa do KDD, foi escolhida a tarefa de *Data Mining* apropriada para o projeto, levando em conta os objetivos definidos. A escolha foi a tarefa de classificação, por ser uma das técnicas mais utilizadas quando deseja-se predizer a classe de novos registros.

Após a escolha da tarefa de *Data Mining*, na sexta etapa, foi decidido o algoritmo a ser utilizado. Foram escolhidos algoritmos de árvores de decisão (ver seção 2.8.1.1) e redes bayesianas (ver seção 2.8.1.2) para a simulação. Redes neurais (ver seção 2.8.1.3) não foram consideradas devido à baixa interpretabilidade e demora para efetuar o treinamento. Todos os algoritmos de classificação consideram registros rotulados com a classe, logo, pode-se afirmar que o aprendizado realizado é supervisionado.

A sétima e a oitava etapa de implementação do algoritmo e avaliação do *Data Mining* foram realizadas na seção 4.4 deste trabalho, de desenvolvimento da proposta.

Ao estudar propostas semelhantes para outros contextos da sociedade, mencionadas no capítulo 3, adotou-se a ideia de gerar uma pontuação para a probabilidade da fraude. Dessa forma, fica registrado se houve alguma emissão de certificado mesmo quando esta foi classificada com uma probabilidade de ser fraude.

4.2 ADAPTAÇÃO DO FLUXO DE ATIVIDADES PARA EMISSÃO DE CERTIFICADO

O fluxo do processo de detecção de fraudes será inserido no fluxo atual de emissão de certificado, logo após a submissão dos dados da validação presencial e antes da emissão do certificado em si. Para que,

baseado na probabilidade calculada pelo classificador, a AR possa tomar a providência que julgar mais adequada.

A Figura 10, a seguir, ilustra o fluxo de detecção de fraudes como parte do fluxo atual de emissão de um certificado:

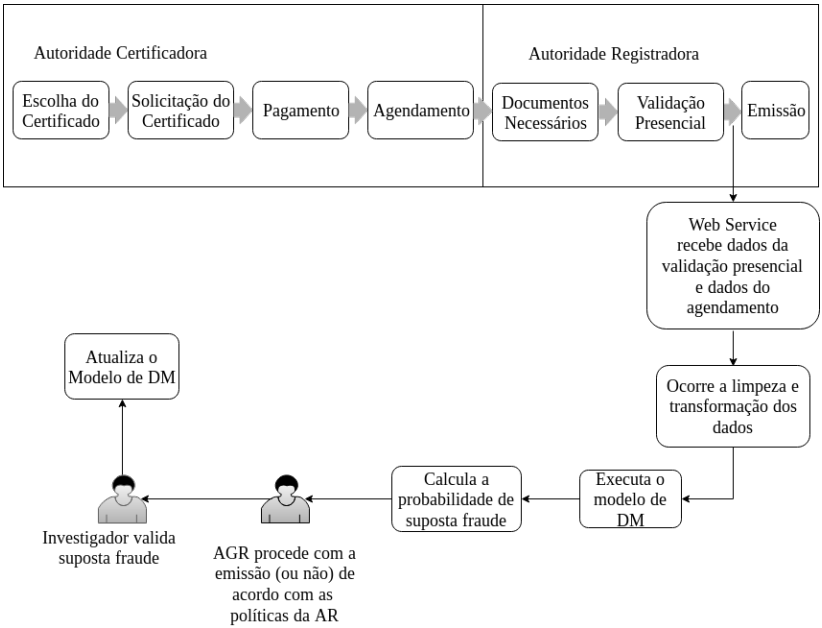

Figura 10 – Fluxo da proposta para predição de fraudes

Dado um cenário em que um indivíduo faça a solicitação de emissão do certificado no sistema da AC, após submetidos os dados da validação presencial, estes dados, juntamente com os dados oriundos do agendamento, são enviados ao Serviço de Detecção de Fraudes (SDF) para que seja feito o cálculo da probabilidade de não ser uma solicitação legítima.

O SDF tem a função de tratar os dados do agendamento do indivíduo e os dados da validação presencial. É nesta etapa que ocorre a adaptação da entrada para que seja compatível ao modelo de classificação.

Após isso, o SDF tratará os dados para a classificação. Várias propostas de outros autores consideram utilizar uma pontuação para o risco de fraude. Após a execução da classificação, essa pontuação é calculada e retornada para o sistema da AR.

Nesta etapa, cada AR pode ter uma política de como proceder de acordo com a pontuação obtida. Caso haja grande chance de ser fraude, a AR pode optar por não emitir o certificado, por exemplo. Além da possibilidade de prosseguir com uma investigação.

Caso a AR decida prosseguir com uma investigação, o investigador valida os motivos pelos quais a pontuação acusou grande chance de fraude, e determina se a classificação está correta ou não. Isso é importante para identificação de falsos positivos, ou seja, ocorrências que tenham sido classificadas como fraudes, mas que sejam ocorrências legítimas.

Com a comprovação de validações presenciais fraudulentas, assim como a classificação de validações legítimas, o modelo de classificação é atualizado.

4.3 PROPOSTAS PARA INTEGRAÇÃO

Após a especificação do fluxo de atividades acima, é necessário elencar as possíveis formas de integrar este modelo com os sistemas das Autoridades já existentes. O ITI não exige que as Autoridades utilizem softwares específicos para o gerenciamento das atividades que envolvem certificação digital. Assim, duas propostas de integração foram definidas a seguir.

4.3.1 Proposta 1: Múltiplas Autoridades de Registro comunicando-se diretamente com o SDF

Esta proposta conta com a comunicação direta dos sistemas já existentes nas ARs com o SDF. Como garantia de autenticidade das partes da comunicação, sugere-se o uso de autenticação mútua.

Esta proposta foi elaborada tendo em mente a simplicidade na implementação. A Figura 11, a seguir, ilustra a arquitetura proposta, seguida de seu fluxo:

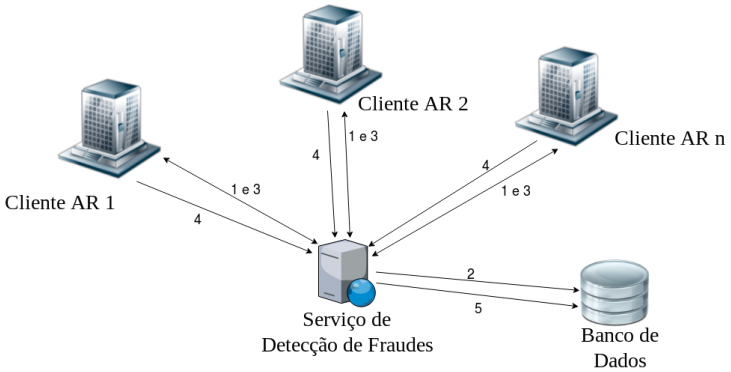

Figura 11 – Arquitetura da proposta para predição de fraudes

1. A AR envia os dados relacionados ao agendamento e à validação presencial para o SDF;

2. O SDF persiste a entrada não classificada;

3. O SDF informa à AR a probabilidade de ser fraude;

4. A AR envia o identificador da ocorrência, juntamente com um indicador se esta foi considerada fraude ou legítima;

5. O SDF atualiza a entrada no Banco de Dados desta ocorrência, agora classificada, e sinaliza o modelo como desatualizado (a atualização do modelo de classificação ocorre diariamente à 1h da manhã).

4.3.2 Proposta 2: Múltiplas Autoridades de Registro comunicando-se com o SDF através de um proxy

Esta proposta conta com a comunicação dos sistemas já existentes nas ARs com um proxy executando em ambiente interno à AR que, por sua vez, comunica-se com o SDF. Devido ao fato de o Proxy estar executando no ambiente da AR, o nível de segurança na comunicação entre este e o sistema da AR fica a critério da própria AR. Já no que diz respeito à comunicação do Proxy com o SDF, sugere-se, também, o uso de autenticação mútua.

O desenvolvimento do Proxy faz parte da proposta, portanto, sua implementação não fica a cargo da AR. Esta proposta foi elaborada

tendo em mente maior facilidade para implantação no ambiente da AR. A Figura 12, a seguir, ilustra a arquitetura proposta, seguida de seu fluxo:

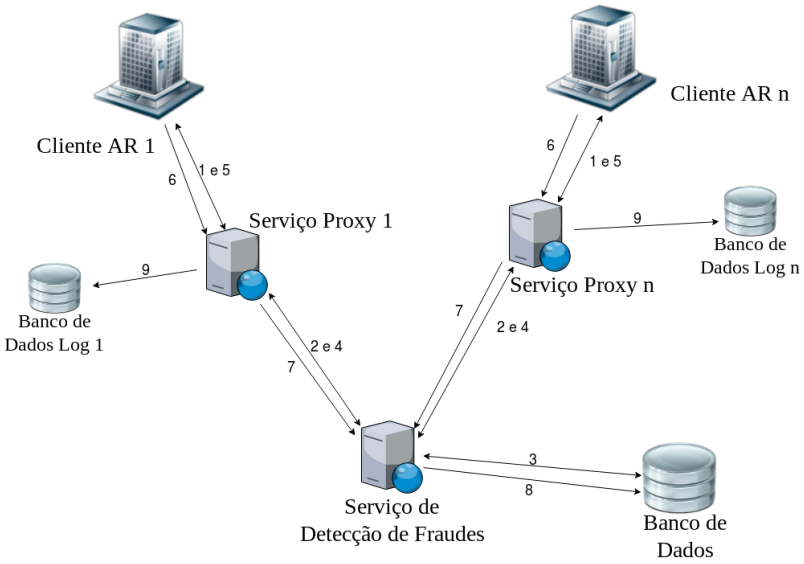

Figura 12 – Arquitetura da proposta para predição de fraudes

1. A AR envia os dados relacionados ao Agendamento e à validação presencial para o Proxy;
2. O Proxy encaminha os dados para o SDF;
3. O SDF persiste a entrada não classificada;
4. O SDF informa ao Proxy a probabilidade de ser Fraude;
5. O Proxy encaminha à AR a probabilidade;
6. A AR envia o identificador da ocorrência, juntamente com um indicador se esta foi considerada fraude ou legítima ao Proxy;
7. O Proxy encaminha os dados para o SDF;
8. O SDF atualiza a entrada no Banco de Dados desta ocorrência, agora classificada, e sinaliza o modelo como desatualizado (a atualização do modelo de classificação ocorre diariamente à 1h da manhã);

9. O Proxy armazena numa base de dados de registros de operações o identificador do usuário, o indicador se foi considerada fraude ou não, o identificador da entrada e sua probabilidade de fraude.

4.4 DESENVOLVIMENTO DA PROPOSTA

A proposta escolhida no desenvolvimento foi a proposta 2, descrita na seção 4.3.2, que conta com o sistema já existente na AR, o SDF e um Proxy para realização da ponte de comunicação de forma segura entre os dois sistemas anteriores. Para esta arquitetura, foram desenvolvidos 3 projetos: *fraud-classifier*, *fraud-classifier-proxy* e *fraud-detector*. Todos foram construídos utilizando a linguagem de programação Java, *Java Persistence API* (JPA), o motor de banco de dados MongoDB, o *framework* Spring, a ferramenta Apache Maven para gerenciamento de dependências, e o software Git para controle de versão e gerenciamento de alterações de código.

A linguagem de programação Java é uma linguagem orientada a objetos escrita na década de 1990 por um grupo de engenheiros da *Sun MicroSystems*. De acordo com Oracle (2017a), o pequeno grupo de engenheiros denominados de "*Green Team*", liderados por James Gosling, desenvolveu o Java com o objetivo de que fosse uma linguagem de cunho didático.

JPA é um *framework* utilizado para persistir objetos do Java no Banco de Dados, servindo como uma interface. Este *framework* fornece uma facilidade no mapeamento objeto/relacional para gerenciar dados relacionais em aplicativos Java (ORACLE, 2017b).

Por causa da grande quantia de dados que espera-se que seja gerada, o MongoDB foi escolhido como banco de dados. Este, conforme descrito em mongoDB (2017), é um banco de dados com escalabilidade e flexibilidade que armazena documentos semelhantes a *JavaScript Object Notation* (JSON). É um banco de dados não relacional, que não leva em consideração entidades (e suas estruturas), relacionamentos e transações.

O Spring, como definido por Weissmann (2014), é um *framework* de programação de aplicações *web* em grande escala, escrito em Java e que possui, como principal característica, a clara divisão de "responsabilidades"entre os diversos componentes de programação que compõem uma aplicação *web*. Também foi utilizado o Spring Boot, que Spring (2017) define como um módulo do Spring que facilita a criação de projetos, fornecendo funcionalidades interessantes, como servidores

embutidos, questões de segurança, entre outras estruturas basicamente pré-configuradas para reduzir o tempo do desenvolvedor com questões de configurações.

O Apache Maven, como descrito por Apache (2017), é uma ferramenta com o objetivo de simplifiar o processo de construção de um projeto, facilitando a compilação, auxiliando em um projeto de qualidade com listas de dependências e correspondências, auxilia no gerenciamento de *releases*, entre outras funcionalidades.

Para o gerenciamento dos códigos fonte do projeto, foi utilizado o software *open source* Git. Ele permite que diferentes pessoas possam participar do desenvolvimento do projeto de forma simultânea. O Git possibilita que os usuários criem, usem e alternem entre "ramos"do projeto, que são chamados de *branches*. Além disso, possibilita que sejam feitas alterações, e estas sejam gerenciadas e compartilhadas a qualquer nível de detalhe (AHMADPANAH, 2015).

4.4.1 Serviço de Detecção de Fraudes

O projeto *fraud-classifier* representa o módulo SDF da arquitetura e consiste de um *web service* para classificação das solicitações de emissão de certificados digitais.

Um *web service* é uma *Application Programming Interface* (API) que, segundo Pautasso (2009), permite comunicação entre aplicações diferentes e disponibiliza seus recursos na rede. REST é a sigla para *Representational State Transfer* (Transferência de Estado Representacional, em tradução livre) e é o termo usado para descrever a arquitetura de API normalmente adotada em *web services*. Masse (2011) descreve uma API REST como a "face"de um *web service*, programada com o objetivo de ser acessível por outras aplicações *web*.

Para melhor compreensão do funcionamento do projeto, e também para futuras implantações do mesmo, foi desenvolvido um manual (ver Anexo A). O projeto possui os seguintes serviços:

- **Ação:** Classificar uma entrada baseado em dados na solicitação de emissão.

  URL: /classify

  Método: POST

  Parâmetros da requisição: (* um dos campos obrigatório / ** campo opcional)

  {

```
    foto: byte[],
    imagemDocumento: byte[],
    assinatura: byte[],
    assinaturaDocumento: byte[],
    digital: byte[],
    digitalDocumento: byte[],
    tipo: string,
    cidadeSolicitante: string,
    ufSolicitante: string,
    razaoSocialAc: string,
    razaoSocialAr: string,
    dataNascimento: date,
    * dataExpedicaoRg: date,
    * dataExpedicaoCnh: date,
    * dataExpedicaoCne: date,
    * dataExpedicaoPassaporte: date,
    * dataExpedicaoCCProfissional: date,
    * dataExpedicaoCtps: date,
    ** dataExpedicaoCnpj: date,
    ** dataExpedicaoCertidao: date
}
```

Códigos de resposta: OK (200) | Bad Request (400)

Corpo da resposta:

```
{
    id: string,
    probabilidadeFraude: double,
    probabilidadeAutentica: double
}
```

- **Ação:** Definir a classe de uma entrada já existente.

  URL: /classify/:id?f=:fraude

  Método: POST

  Parâmetros da URL:

  ```
  id: string
  fraude: boolean
  ```

  Códigos de resposta: OK (200) | Bad Request (400)

  Corpo da resposta:

  ```
  {
      id: string,
  ```

```
    probabilidadeFraude: double,
    probabilidadeAutentica: double
}
```

Ao submeter uma solicitação de emissão ao classificador, esta entrada é persistida no banco de dados, classificada e um objeto representando a classificação da mesma é retornado como resposta. Posteriormente, essa entrada pode ter sua classe definida, de forma que ela faça parte de um modelo de classificação futuro. Nem todas as entradas serão classificadas, uma vez que a emissão do certificado pode ser cancelada sem que necessariamente a causa seja o fato de tratar-se de uma fraude.

Dentre os dados armazenados no banco de dados do SDF, estão: as entradas propriamente ditas, que representam as solicitações de emissão de certificado classificadas ou não; e os dias não úteis (exceto fins de semana), usados na validação das datas de emissão dos documentos. A seguir, é descrita a modelagem dos objetos armazenados neste banco de dados:

- **Entidade:** Entrada

  Descrição: Representação de uma solicitação de emissão de certificado digital.

  Estrutura:

```
{
    id: org.bson.types.ObjectId,
    dataNascimento: java.util.Date,
    similaridadeFotos: java.lang.Double,
    similaridadeAssinaturas: java.lang.Double,
    similaridadeDigitais: java.lang.Double,
    dataExpedicaoDiaUtilRg: java.lang.Boolean,
    dataExpedicaoDiaUtilCnh: java.lang.Boolean,
    dataExpedicaoDiaUtilCne: java.lang.Boolean,
    dataExpedicaoDiaUtilPassaporte: java.lang.Boolean,
    dataExpedicaoDiaUtilCCProfissional: java.lang.Boolean,
    dataExpedicaoDiaUtilCtps: java.lang.Boolean,
    dataExpedicaoDiaUtilCnpj: java.lang.Boolean,
    dataExpedicaoDiaUtilCertidao: java.lang.Boolean,
    tipo: java.lang.String,
    cidadeSolicitante: java.lang.String,
    ufSolicitante: java.lang.String,
    razaoSocialAc: java.lang.String,
    razaoSocialAr: java.lang.String,
```

```
    probabilidadeFraude: java.lang.Double,
    probabilidadeAutentica: java.lang.Double
    fraude: java.lang.Boolean
}
```

- **Entidade:** DiaNaoUtil

  Descrição: Representação de um dia não útil, utilizado na classificação para determinar se as datas de emissão dos documentos ocorrem em dias úteis ou não. Não são armazenados dias correspondentes a fins de semana, uma vez que estes podem ser calculados diretamente no código.

  Estrutura:

```
{
    id: org.bson.types.ObjectId,
    data: java.util.Date,
    nome: java.lang.String
}
```

A seção 4.1 menciona o uso de medidas de similaridade entre imagens contendo o rosto, assinatura e digital do requerente. No entanto, a implementação de tais métodos de cálculo de similaridade entre imagens foi considerado um item fora do escopo deste trabalho como um trabalho de conclusão de curso, uma vez que, de acordo com a seção 1.2, este trata-se de uma proposta de um modelo para predição de fraudes com esforços focados em segurança e ciência de dados, não na implementação de um sistema que realize um tratamento de imagens de tamanha complexidade. Por isso, foi utilizada uma determinação de similaridade pseudoaleatória entre duas imagens. A função de determinação de similaridade tem probabilidades de 80% de gerar um valor no intervalo de *[0.6; 1]* e 20% de gerar um valor no intervalo de *[0; 0.6]*. Isso confere ao sistema uma proporção considerada mais "realista"para a natureza dos possíveis resultados de um cálculo de similaridade entre essas categorias de imagens.

Para a predição da probabilidade de uma entrada caracterizar uma fraude ou não, foi utilizada a API para Java do Weka (WITTEN et al., 2016). O Weka é um software desenvolvido em Java pelo grupo de pesquisa de *Machine Learning* da Universidade de Waikato, da Nova Zelândia. Este software conta com uma API munida de diversos algoritmos e utilidades para a classificação de dados utilizando *Data Mining*. O algoritmo de classificação pode ser definido pelo administrador do sistema por meio de parâmetros na execução do mesmo. Caso ele queira

visualizar estatísticas como acurácia ou precisão (ver seção 2.8.1.6), estas são acessíveis por meio do seguinte serviço (disponível apenas para administradores):

- **Ação:** Visualizar estatísticas do classificador.

  URL: /evaluate?folds=:folds&arff=:gerarArff

  Método: GET

  Parâmetros da URL: (* campo opcional)

  ```
  * folds: integer  (10)
  * gerarArff: boolean  (false)
  ```

  Corpo da resposta: Representação textual da estrutura de classificação criada e resumo das estatísticas do classificador em execução.

  Códigos de resposta: OK (200)

Por questões de auditoria, todas as operações realizadas no *fraud-classifier* (inserções, atualizações e classificações) são registradas num arquivo de registros do servidor de aplicação.

4.4.2 Proxy

O projeto *fraud-classifier-proxy* representa o módulo Proxy da arquitetura e tem como principal função realizar a ponte de comunicação entre o sistema da AR e o SDF utilizando a autenticação mútua. Este também é um *web service* com a arquitetura REST.

Para melhor compreensão do funcionamento do projeto, e também para futuras implantações do mesmo, foi desenvolvido um manual (ver Anexo B). Este projeto possui serviços semelhantes ao *fraud-classifier*:

- **Ação:** Redirecionar a entrada com os dados da solicitação de emissão para o SDF classificar.

  URL: /classify

  Método: POST

  Parâmetros da requisição: (* um dos campos obrigatório / ** campo opcional)

  ```
  {
     foto: byte[],
  ```

```
    imagemDocumento: byte[],
    assinatura: byte[],
    assinaturaDocumento: byte[],
    digital: byte[],
    digitalDocumento: byte[],
    tipo: string,
    cidadeSolicitante: string,
    ufSolicitante: string,
    razaoSocialAc: string,
    razaoSocialAr: string,
    dataNascimento: date,
    * dataExpedicaoRg: date,
    * dataExpedicaoCnh: date,
    * dataExpedicaoCne: date,
    * dataExpedicaoPassaporte: date,
    * dataExpedicaoCCProfissional: date,
    * dataExpedicaoCtps: date,
    ** dataExpedicaoCnpj: date,
    ** dataExpedicaoCertidao: date
}
```

Códigos de resposta: OK (200) | Bad Request (400)

Corpo da resposta:

```
{
    id: string,
    probabilidadeFraude: double,
    probabilidadeAutentica: double
}
```

- **Ação:** Registrar a definição da classe de uma entrada por determinado usuário da AR e redirecionar para o SDF atualizar a classe desta entrada.

  URL: /classify/:id?f=:fraude

  Método: PUT

  Parâmetros da URL:

  ```
  id: string
  fraude: boolean
  ```

  Parâmetros da requisição:

  ```
  idUsuario: long
  ```

Códigos de resposta: OK (200) | Bad Request (400)

De maneira semelhante ao SDF, as operações são registradas, entretanto, não no arquivo de registros do servidor de aplicação, mas num banco de dados. Uma vez que o Proxy executa em um ambiente interno à AR, é possível armazenar dados mais sensíveis como os identificadores dos usuários que realizaram determinadas ações. Desta forma, numa auditoria, é possível identificar qual usuário aprovou a emissão de um certificado para uma solicitação de autenticidade duvidosa, por exemplo. A descrição da modelagem dos registros das operações é feita a seguir:

- **Entidade:** Log

  Descrição: Representação de uma ação de aprovação ou não de uma entrada armazenada no banco de dados

  Estrutura:

```
{
    id: org.bson.types.ObjectId,
    idEntrada: org.bson.types.ObjectId,
    idUsuario: java.lang.Long,
    aprovada: boolean
    probabilidadeFraude: java.lang.Double,
    probabilidadeAutentica: java.lang.Double
}
```

Uma vez que este projeto tem o objetivo de ser uma ponte de comunicação segura entre o sistema da AR e o SDF, o ponto mais importante deste módulo é o uso de chaves que serão utilizadas na autenticação mútua. Para isto, foram gerados certificados digitais para o SDF e para o Proxy que "confiam"um no outro. Para a geração destes certificados, foi realizado o seguinte procedimento:

1. Geração de certificado autoassinado para o SDF e para o Proxy:

```
$ keytool -genkey -alias server -keystore sdf.p12 -keyalg
    RSA -storetype PKCS12
$ keytool -genkey -alias client -keystore proxy.p12
    -keyalg RSA -storetype PKCS12
```

2. Exportação dos certificados

```
$ keytool -export -alias server -file sdf.cer -keystore
    sdf.p12
```

```
$ keytool -export -alias client -file proxy.cer -keystore
    proxy.p12
```

3. Importação dos certificados do SDF e do Proxy nas áreas de armazenamento de chaves um do outro

```
$ keytool -importcert -file proxy.cer -keystore sdf.p12
    -alias client
$ keytool -importcert -file sdf.cer -keystore proxy.p12
    -alias server
```

Posteriormente, estes certificados foram importados e referenciados nos respectivos projetos para serem utilizados na comunicação. O transporte dos certificados não foi definido na proposta, mas deve ser realizado de uma forma segura para que não haja risco de interceptação da mesma por um terceiro mal-intencionado.

4.4.3 Simulação com Autoridade Registradora

No primeiro momento, foi necessário criar uma base de dados fictícia para treinar os modelos. Após isso, realizou-se a simulação do fluxo de uma emissão de certificado digital, incluindo a adaptação da proposta.

4.4.3.1 Geração de dados fictícios

Como descrito nas limitações desse trabalho, não foi possível obter dados reais deste banco de dados. Para isso foram criados dados fictícios com o objetivo de simular a execução do modelo de classificação. Phua et al. (2010) comenta em seu artigo, que esta é uma alternativa válida quando se trata de fraudes, pela dificuldade de obter dados reais.

Foi utilizado um conjunto de 12 cidades de 3 estados diferentes (Florianópolis, Palhoça, São José e Videira de Santa Catarina; Porto Alegre, Pelotas e Caxias do Sul do Rio Grande do Sul; e São Paulo, Itú, São Caetano do Sul, Bragança Paulista e Limeira de São Paulo), 3 políticas de certificado (A1, A2 e A3), 4 ACs e 6 ARs fictícias. Para cada um dos 50 mil registros gerados, é sorteada uma AR (que é associada a uma AC e uma cidade) e uma das 3 políticas de certificado para ser associada ao registro. Além disso, é gerada uma data de nas-

cimento e uma data de expedição para o RG com valores posteriores a 1º de janeiro de 2001. Esse limite mínimo para datas ocorre devido aos feriados importados na base de dados, que são todos posteriores a esta data. Após a geração das datas de expedição do RG, baseado numa chance de 95% de existência de cada documento de identificação dos mencionados na seção 4.1 e do CNPJ, é gerada uma data de expedição destes, posterior à data de expedição do RG e, baseado numa chance de 90% de existência da certidão de nascimento, é gerada uma data de expedição da certidão de nascimento posterior à data de nascimento. Lembrando que, conforme mencionado na seção 4.4.1, as medidas de similaridades serão geradas no momento em que cada solicitação é tratada antes de ser persistida no banco de dados. Após a entrada ser armazenada, esta deve ter sua classe definida. Para tal, é verificado se alguma das medidas de similaridade está abaixo de um limite definido (0.4) ou se alguma das datas geradas ocorre em algum dia não útil. Caso nenhuma destas condições seja satisfeita, ainda existe uma chance de 10% que esta entrada seja definida como fraude. Com isso, tem-se uma coleção de registros considerados suficientes para os testes das aplicações.

A seguir, são mostradas as distribuições dos atributos para o conjunto gerado com o método descrito acima. Observando as Figuras 13, 14 e 15, que se tratam do valor de similaridade, é possível constatar uma relação direta com as entradas que apresentam um baixo grau de similaridade (em qualquer um dos 3 atributos) com a ocorrência de fraude. Isto se deve ao fato de os valores da similaridade terem sido utilizados como fatores determinantes para definição da entrada como uma fraude ou não durante a geração dos dados. Nos casos em que o grau de similaridade é mais elevado, um outro atributo serviu como fator para a categorização das entradas como fraudes.

Figura 13 – Gráfico gerado no Weka sobre o atributo **similaridadeFotos**

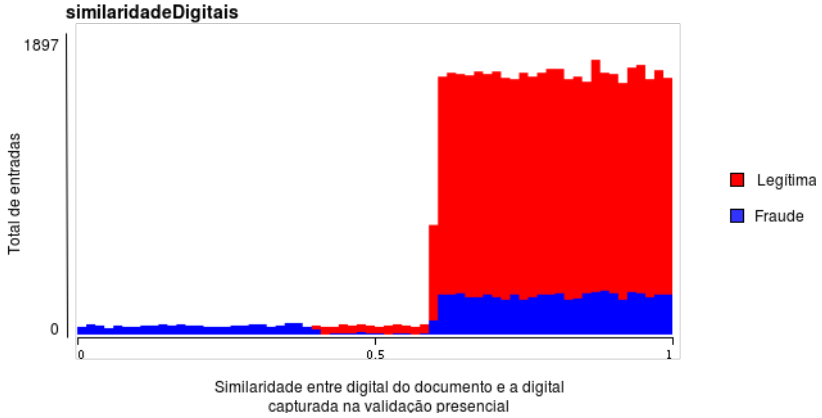

Figura 15 – Gráfico gerado no Weka sobre o atributo **similaridadeDigitais**

Na Figura 16, a seguir, é mostrada a distribuição das datas de nascimento das entradas. Neste caso não é possível relacionar a data de nascimento com a classe das entradas.

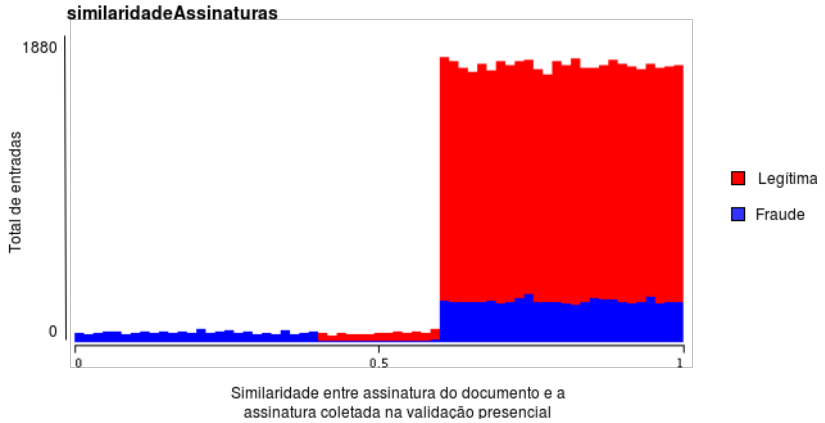

Figura 14 – Gráfico gerado no Weka sobre o atributo **similaridadeAssinaturas**

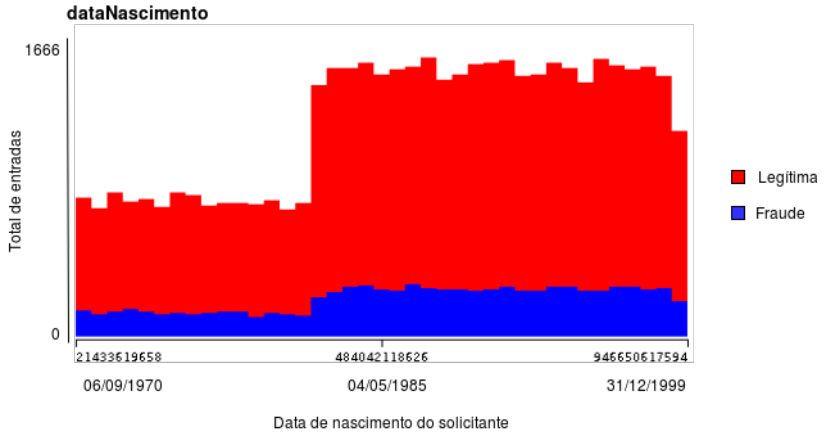

Figura 16 – Gráfico gerado no Weka sobre o atributo **dataNascimento**

Observando as Figuras 17, 18, 19, 20, 21, 22, 23 e 24 é possível constatar que, quando a data de emissão de algum documento não é dia útil, isso caracteriza a entrada como fraude. Já nos casos em que a data é um dia útil, é possível inferir que outro atributo (possivelmente uma outra data) determina a fraude, já que a maioria das entradas é definida como legítima neste caso.

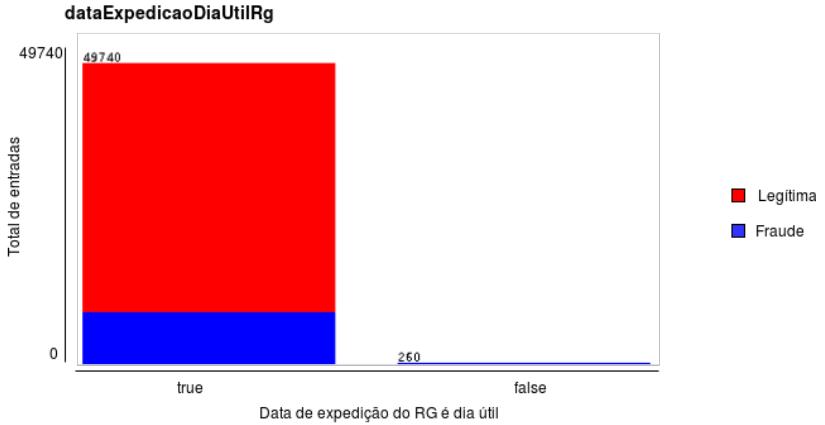

Figura 17 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilRg**

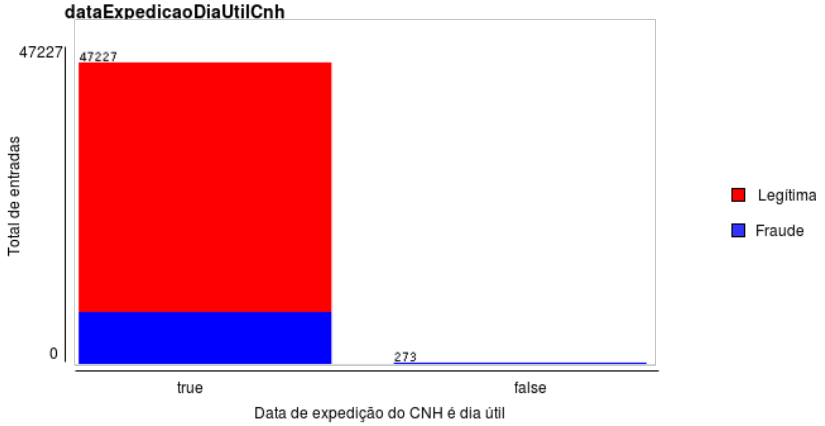

Figura 18 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCnh**

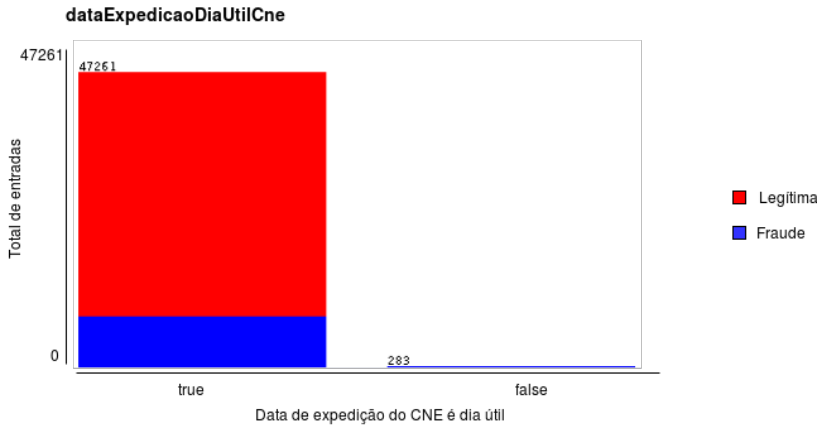

Figura 19 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCne**

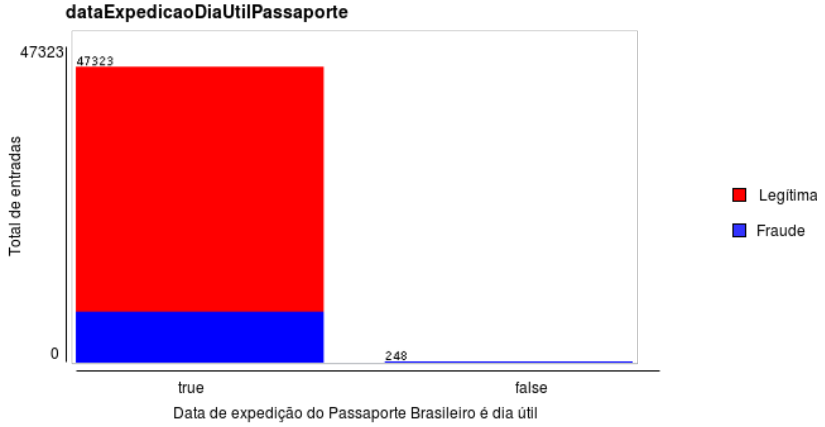

Figura 20 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilPassaporte**

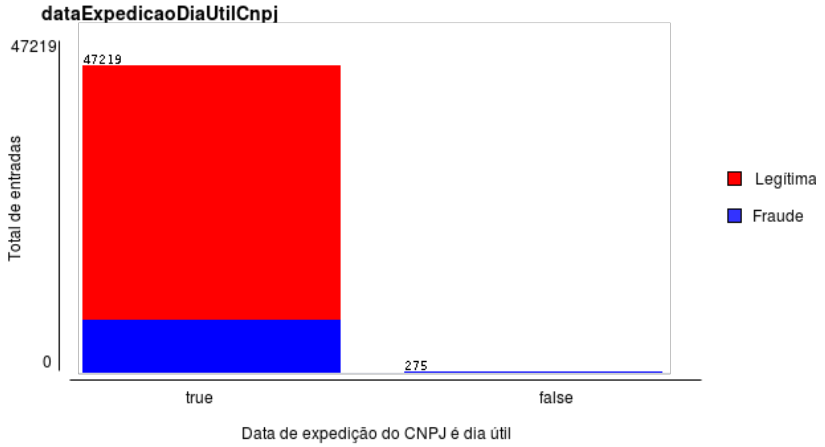

Figura 21 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCnpj**

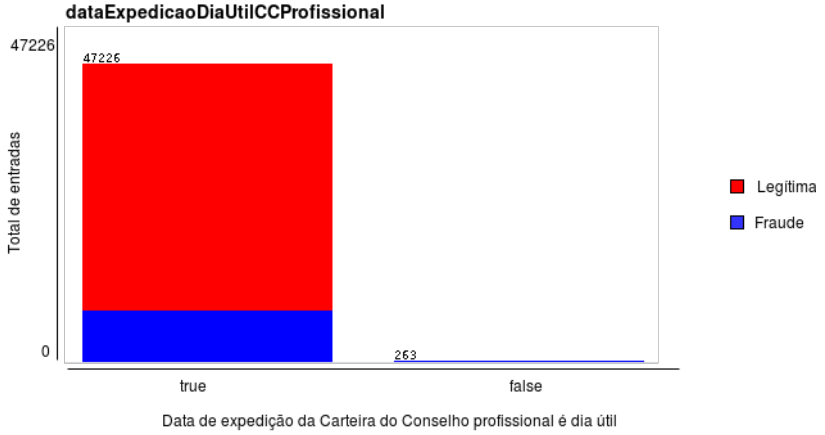

Figura 22 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCCProfissional**

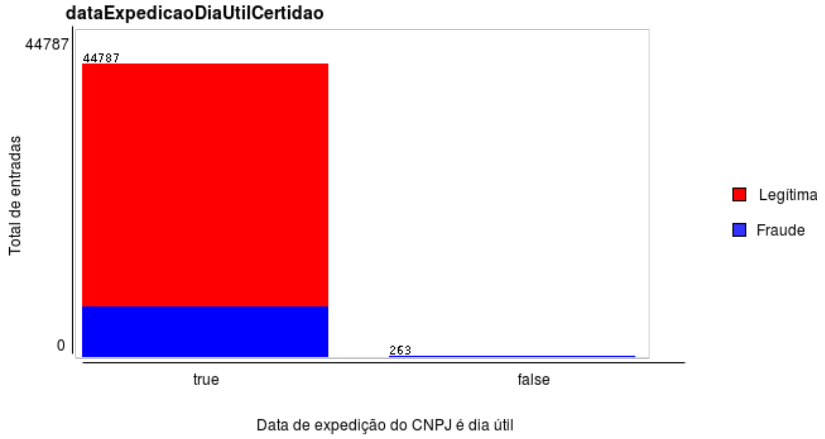

Figura 23 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCertidao**

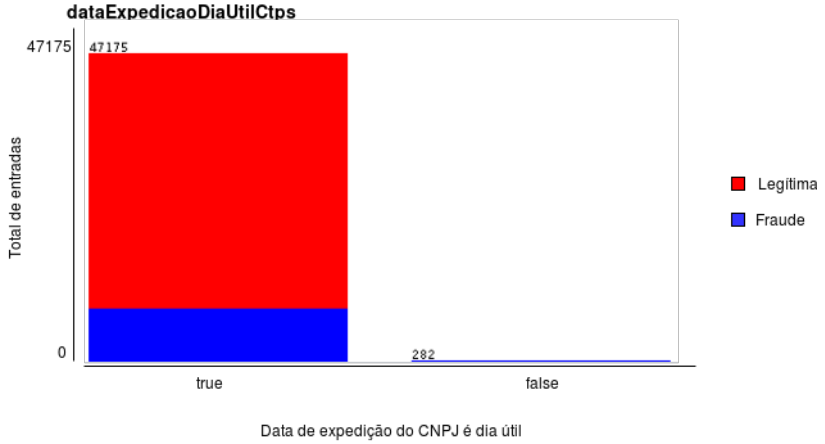

Figura 24 – Gráfico gerado no Weka sobre o atributo **dataExpedicaoDiaUtilCtps**

As Figuras 25, 26, 27 e 28 mostram a distribuição dos valores referentes aos índices de ocorrência de fraudes por cidade e UF de residência do requerente, razão social da AC e razão social da AR mencionados na seção 4.1. As imagens mostram pouca distribuição dos dados dentre os índices e uma ocorrência de fraude sempre proporcional, não

importando a quantidade de entradas para cada valor.

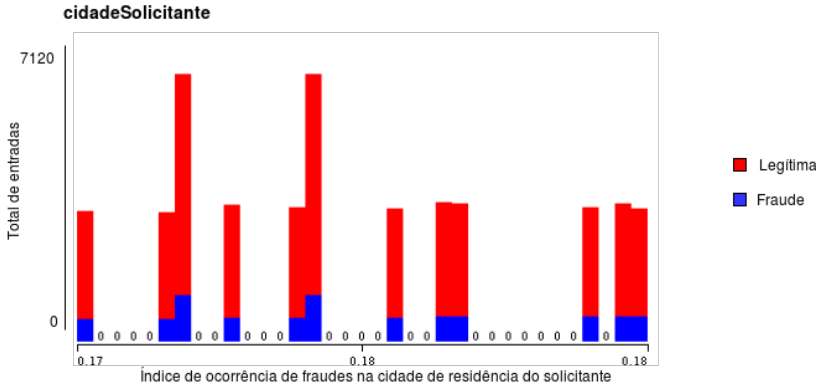

Figura 25 – Gráfico gerado no Weka sobre o atributo **cidadeSolicitante**

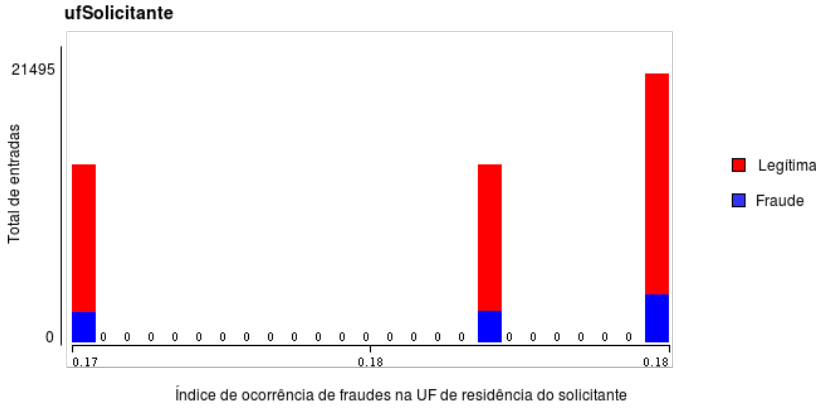

Figura 26 – Gráfico gerado no Weka sobre o atributo **ufSolicitante**

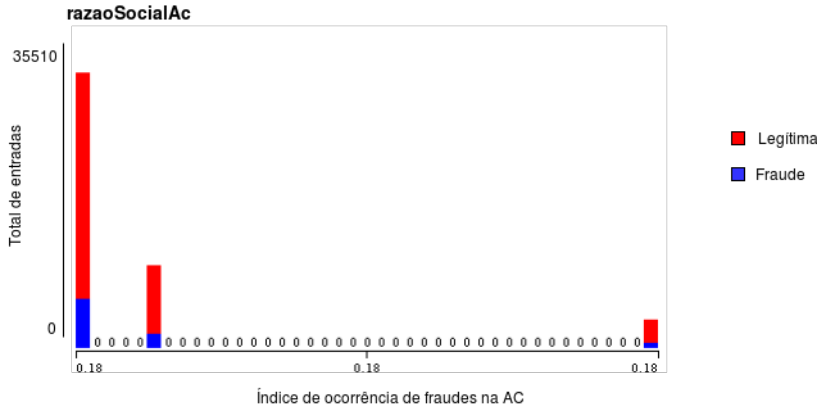

Figura 27 – Gráfico gerado no Weka sobre o atributo **razaoSocialAc**

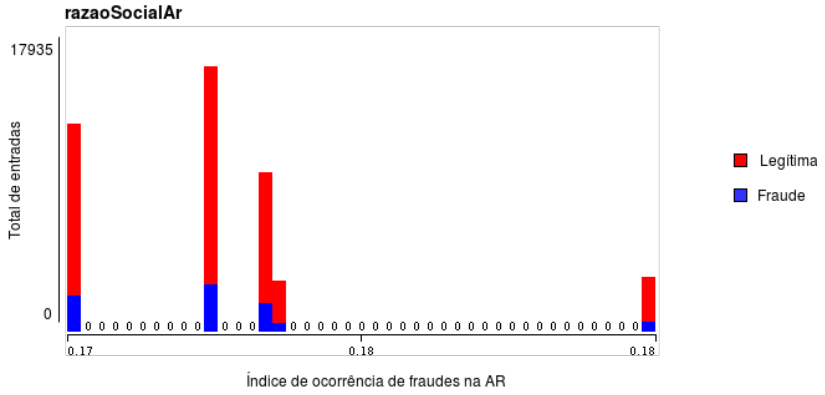

Figura 28 – Gráfico gerado no Weka sobre o atributo **razaoSocialAr**

Uma vez gerados os dados, foram executados alguns algoritmos de classificação dos mencionados na seção 2.8.1. As Figuras 29 e 30 mostram as árvores de decisão (ver seção 2.8.1.1) criadas pelos classificadores J48 (obtendo uma acurácia de 95,66%) e *Hoeffding Tree* (obtendo uma acurácia de 94,52%).

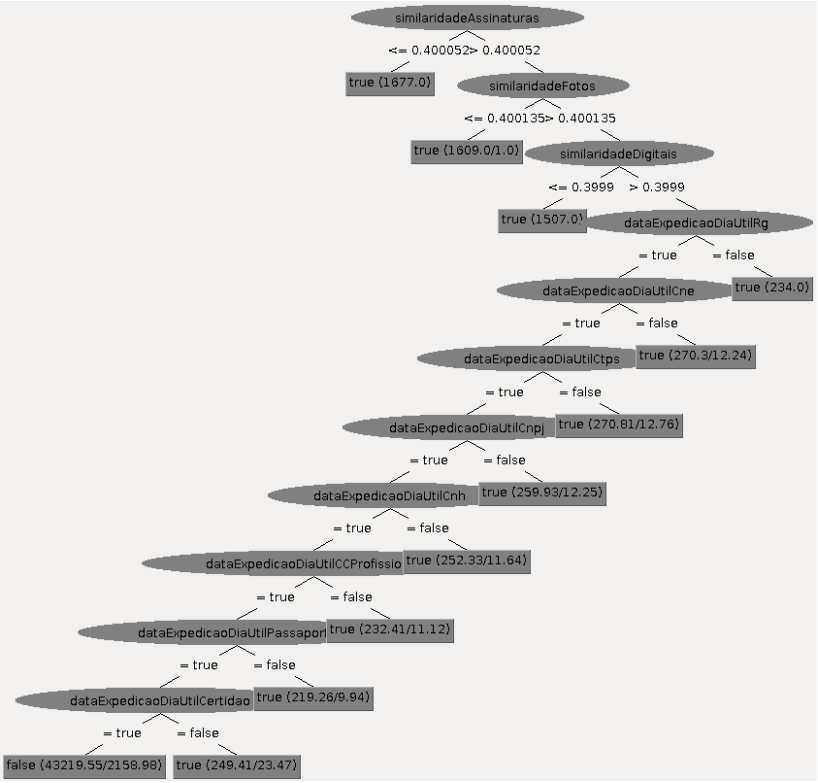

Figura 29 – Árvore de decisão gerada pelo classificador J48

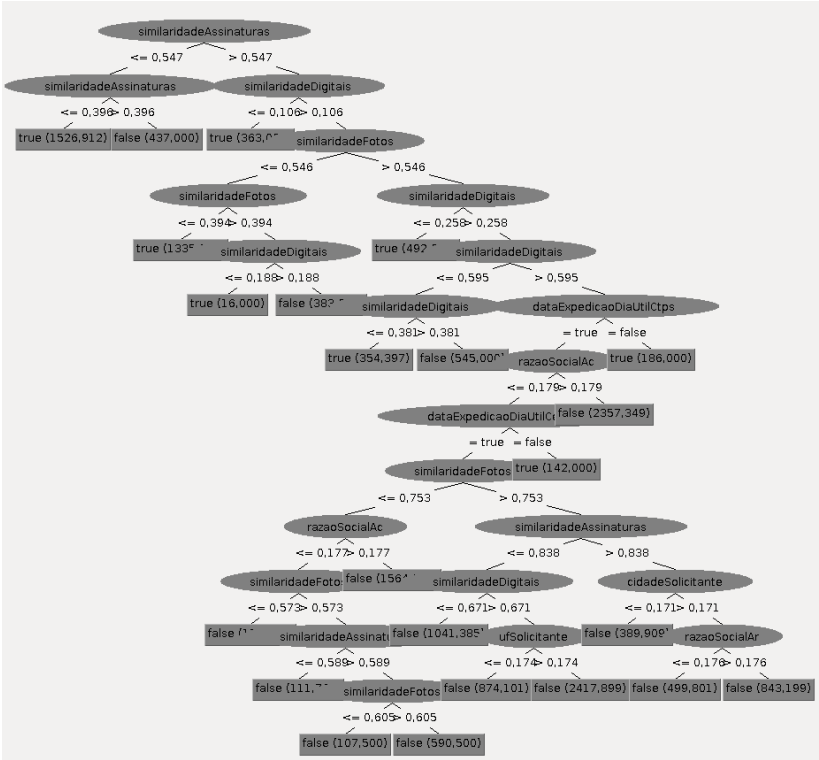

Figura 30 – Árvore de decisão gerada pelo classificador *Hoeffding Tree*

As Figuras 31 e 32 representam o modelo gerado pela rede bayesiana (ver seção 2.8.1.2) do classificador *Naive Bayes*, que obteve uma acurácia de 94,56%.

```
Naive Bayes Classifier

                                          Class
Attribute                                  true             false
                                         (0.18)           (0.82)
=============================================================================
dataNascimento
  mean                      549409345477.2375  547717144353.866
  std. dev.                 249231042566.8931 247969796366.4785
  weight sum                             8846             41155
  precision                  18504339.9587     18504339.9587

similaridadeFotos
  mean                                 0.6842           0.7947
  std. dev.                            0.2631           0.1216
  weight sum                             8846             41155
  precision                                 0                0

similaridadeAssinaturas
  mean                                 0.6821            0.795
  std. dev.                            0.2621           0.1207
  weight sum                             8846             41155
  precision                                 0                0

similaridadeDigitais
  mean                                 0.6882           0.7947
  std. dev.                            0.2587           0.1211
  weight sum                             8846             41155
  precision                                 0                0

tipo
  A3                                   2918.0           13856.0
  A1                                   2946.0           13694.0
  A2                                   2985.0           13608.0
  [total]                              8849.0           41158.0

cidadeSolicitante
  mean                                 0.1769           0.1767
  std. dev.                            0.0048           0.0048
  weight sum                             8846             41155
  precision                            0.0014           0.0014

ufSolicitante
  mean                                 0.1767           0.1766
  std. dev.                            0.0035           0.0035
  weight sum                             8846             41155
  precision                            0.0039           0.0039

razaoSocialAc
  mean                                 0.1765           0.1765
  std. dev.                            0.0022           0.0021
  weight sum                             8846             41155
  precision                            0.0027           0.0027
```

Figura 31 – Primeira parte do modelo gerado pelo classificador *Naive Bayes*

```
razaoSocialAr
  mean                                  0.1771          0.1771
  std. dev.                             0.0024          0.0024
  weight sum                              8846           41155
  precision                              0.002           0.002

dataExpedicaoDiaUtilRg
  true                                  8587.0          41156.0
  false                                  261.0              1.0
  [total]                               8848.0          41157.0

dataExpedicaoDiaUtilCnh
  true                                  8114.0          39116.0
  false                                  274.0              1.0
  [total]                               8388.0          39117.0

dataExpedicaoDiaUtilCne
  true                                  8145.0          39118.0
  false                                  284.0              1.0
  [total]                               8429.0          39119.0

dataExpedicaoDiaUtilPassaporte
  true                                  8154.0          39171.0
  false                                  249.0              1.0
  [total]                               8403.0          39172.0

dataExpedicaoDiaUtilCtps
  true                                  8129.0          39049.0
  false                                  283.0              1.0
  [total]                               8412.0          39050.0

dataExpedicaoDiaUtilCCProfissional
  true                                  8177.0          39051.0
  false                                  264.0              1.0
  [total]                               8441.0          39052.0

dataExpedicaoDiaUtilCnpj
  true                                  8162.0          39059.0
  false                                  276.0              1.0
  [total]                               8438.0          39060.0

dataExpedicaoDiaUtilCertidao
  true                                  7731.0          37059.0
  false                                  264.0              1.0
  [total]                               7995.0          37060.0
```

Figura 32 – Segunda parte do modelo gerado pelo classificador *Naive Bayes*

A Figura 33 representa o hiperplano (ver seção 2.8.1.4) do classificador SMO, que obteve uma acurácia de 91.84%

```
SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: true, false

BinarySMO

Machine linear: showing attribute weights, not support vectors.

        -0.0122 * (normalized) dataNascimento
 +       3.329  * (normalized) similaridadeFotos
 +       3.4753 * (normalized) similaridadeAssinaturas
 +       3.2905 * (normalized) similaridadeDigitais
 +       0.0075 * (normalized) tipo=A3
 +      -0.005  * (normalized) tipo=A1
 +      -0.0026 * (normalized) tipo=A2
 +      -0.0172 * (normalized) cidadeSolicitante
 +      -0.0045 * (normalized) ufSolicitante
 +       0.0095 * (normalized) razaoSocialAc
 +      -0.0347 * (normalized) razaoSocialAr
 +      -4.1308 * (normalized) dataExpedicaoDiaUtilRg=false
 +      -4.1368 * (normalized) dataExpedicaoDiaUtilCnh=false
 +      -4.1913 * (normalized) dataExpedicaoDiaUtilCne=false
 +      -4.1222 * (normalized) dataExpedicaoDiaUtilPassaporte=false
 +      -4.1899 * (normalized) dataExpedicaoDiaUtilCtps=false
 +      -4.0818 * (normalized) dataExpedicaoDiaUtilCCProfissional=false
 +      -4.1677 * (normalized) dataExpedicaoDiaUtilCnpj=false
 +      -4.089  * (normalized) dataExpedicaoDiaUtilCertidao=false
 -       6.2334

Number of kernel evaluations: 625478173 (26.119% cached)
```

Figura 33 – Descrição dos pesos usados pelo classificador SMO

Com base nesses resultados, definiu-se o J48 como o algoritmo padrão para o SDF. Embora, conforme descrito no manual do SDF, esse parâmetro pode ser alterado de acordo com a vontade do administrador do sistema.

4.4.3.2 Demonstração do fluxo de emissão de um certificado digital adaptado

Após a criação do conjunto de dados, é necessário mostrar o funcionamento dos componentes em meio a um fluxo que simule as operações reais de emissão de certificados digitais para prova de conceito da proposta. O projeto *fraud-detector* representa o sistema utilizado pela

AR e foi desenvolvido com este propósito. Foi desenvolvido um sistema que simule o cadastro e a aprovação (ou não) de solicitações de emissão de certificados digitais.

O projeto simula o uso por usuários com diferentes níveis de acesso, onde usuários de atendimento têm acesso a funções de cadastro e consulta de agendamentos, enquanto agentes têm acesso, também, às funcionalidades de aprovação (ou não) das solicitações de emissão de certificados fazendo uso do sistema de classificação para auxílio na detecção de fraudes. As figuras a seguir são capturas de tela do sistema desenvolvido.

A Figura 34, a seguir, representa a interface em que o AGR efetua o *login*, de acordo com as informações passadas no cadastro previamente feito.

Figura 34 – Login do sistema da AR

Após efetuar o *login*, o AGR tem acesso à interface apresentada na Figura 35. Ele pode visualizar a lista de agendamentos já cadastrados, e também pode adicionar novos agendamentos ou editar os existentes.

Figura 35 – Agendamentos já cadastrados

Caso o AGR opte por um novo agendamento, a interface mostrada na Figura 36 é apresentada para preencher as principais informações do requerente do certificado.

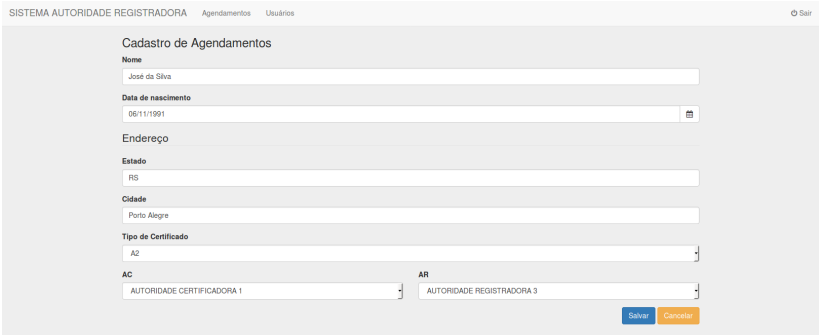

Figura 36 – Formulário para cadastro de novo agendamento

Ao efetuar a validação presencial, o AGR deve abrir um agendamento existente e submeter os dados extras que são exigidos pelo SDF, apresentado na Figura 37 a seguir.

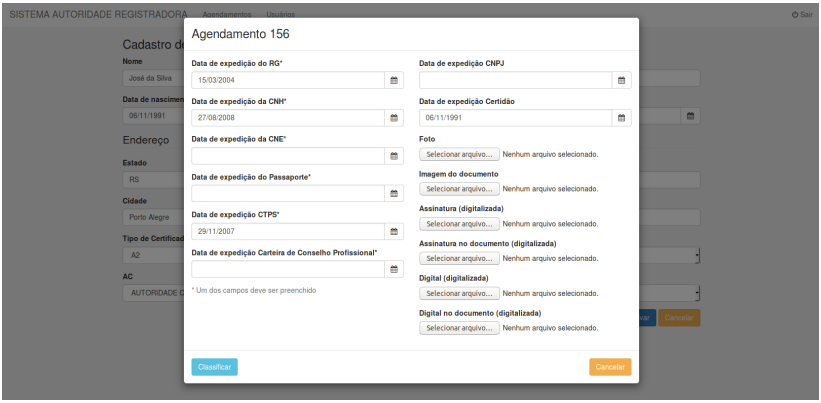

Figura 37 – Formulário para submissão de dados adicionais para classificação da solicitação

Após o envio dos dados e a solicitação de classificação, o SDF retorna a probabilidade de fraude desta nova entrada, como mostra a Figura 38 a seguir.

Figura 38 – Resultado da classificação feita pelo SDF, com as opções **Aprovar** e **Reprovar** habilitadas

Como mencionado anteriormente, cada AR pode ter uma política de como proceder de acordo com o valor de probabilidade de fraude, mas caso seja alta, não é recomendado emitir o certificado. Dessa forma pode-se seguir com os processos definidos pelo ITI para investigar a irregularidade.

Como exemplo de detecção do fraudador externo, foi inserido um indivíduo com um dos seus documentos de identificação expedidos no sábado. Os dados inseridos podem ser observados na figura 39:

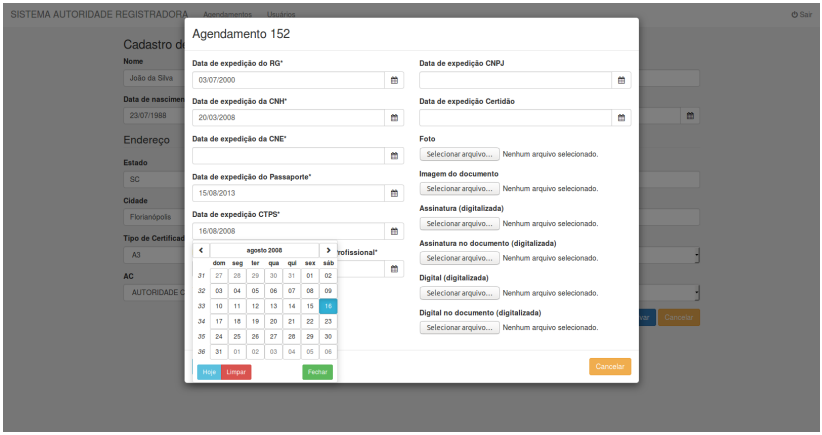

Figura 39 – Formulário para submissão de dados adicionais para classificação da solicitação fraudulenta

O SDF retornou 100% de probabilidade de ser fraude devido ao modelo gerado com o algoritmo J48 que está sendo executado, como mostra a figura 29. Datas de expedição em dias não úteis consideram a solicitação uma fraude. Podemos observar a interface com o resultado da classificação na figura 40:

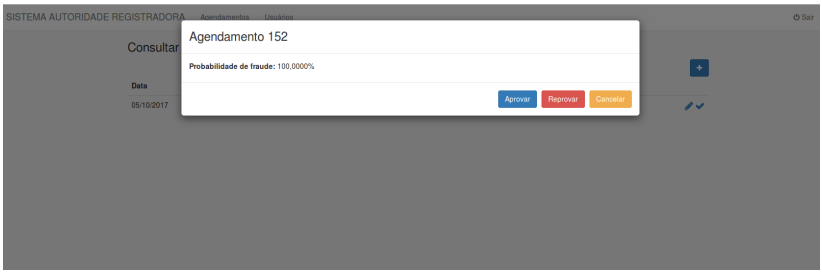

Figura 40 – Resultado da classificação feita pelo SDF para o fraudador externo

As operações realizadas acima resultaram nos registros no log do sistema *fraud-classifier* representados na Figura 41.

```
Persisted Entry [br.ufsc.labsec.fraudclassifier.model.Entrada@243fe552
[id=59d68673dace9716dea3b393,dataNascimento=Wed Nov 06 00:00:00 BRST
1991,similaridadeFotos=0.8146318601112084,similaridadeAssinaturas=0.
6818244922015575,similaridadeDigitais=0.
8961154711503589,tipo=A2,cidadeSolicitante=Porto
Alegre,ufSolicitante=RS,razaoSocialAc=AUTORIDADE CERTIFICADORA
1,razaoSocialAr=AUTORIDADE REGISTRADORA 3,dataExpedicaoDiaUtilRg=true,
dataExpedicaoDiaUtilCnh=true,dataExpedicaoDiaUtilCne=<null>
,dataExpedicaoDiaUtilPassaporte=<null>
,dataExpedicaoDiaUtilCtps=true,dataExpedicaoDiaUtilCCProfissional=<
null>,dataExpedicaoDiaUtilCnpj=<null>
,dataExpedicaoDiaUtilCertidao=true,fraude=<null>,probabilidadeFraude=<
null>,probabilidadeAutentica=<null>]]


Classified Entry [59d68673dace9716dea3b393]: fraude=[true:0.
06983829215650302 / false:0.9301617078436585]


Updated Entry [59d68673dace9716dea3b393]: set fraude=false
```

Figura 41 – Registros do *fraud-classifier*

Onde a entrada no banco de dados pode ser vista na Figura 42.

```
{
    "_id" : ObjectId("59d68673dace9716dea3b393"),
    "_class" : "br.ufsc.labsec.fraudclassifier.model.Entrada",
    "dataNascimento" : ISODate("1991-11-06T00:00:00.000-02:00"),
    "similaridadeFotos" : 0.8146318601112084,
    "similaridadeAssinaturas" : 0.6818244922015575,
    "similaridadeDigitais" : 0.8961154711503589,
    "tipo" : "A2",
    "cidadeSolicitante" : "Porto Alegre",
    "ufSolicitante" : "RS",
    "razaoSocialAc" : "AUTORIDADE CERTIFICADORA 1",
    "razaoSocialAr" : "AUTORIDADE REGISTRADORA 3",
    "dataExpedicaoDiaUtilRg" : true,
    "dataExpedicaoDiaUtilCnh" : true,
    "dataExpedicaoDiaUtilCtps" : true,
    "dataExpedicaoDiaUtilCertidao" : true,
    "fraude" : false,
    "probabilidadeFraude" : 0.06983829215650302,
    "probabilidadeAutentica" : 0.9301617078436585
}
```


Figura 42 – Entrada no banco de dados do *fraud-classifier*

Já no *fraud-classifier-proxy*, a ação ficou registrada com a entrada no banco de dados representada na Figura 43, a seguir.


```
{
    "_id" : ObjectId("59d6868adace9716df68f5ed"),
    "_class" : "br.ufsc.labsec.fraudclassifier.proxy.model.Log",
    "idEntrada" : ObjectId("59d68673dace9716dea3b393"),
    "idUsuario" : NumberLong("1"),
    "aprovada" : true,
    "probabilidadeFraude" : 0.06983829215650302,
    "probabilidadeAutentica" : 0.9301617078436585
}
```


Figura 43 – Registro da ação no banco de dados do *fraud-classifier-proxy*

Dessa forma, para identificar um possível fraudador interno, pode-se utilizar os dados de log que estão sendo persistidos no banco de dados através do projeto *fraud-classifier-proxy*. Por exemplo, com a consulta ilustrada na Figura 44, são retornados os identificadores dos usuários que aprovaram a emissão de certificado para pessoas que obtiveram uma probabilidade de fraude maior que 50%.

```
db.log.find(
    {
        probabilidadeFraude: {
            $gt: 0.5
        },
        aprovada: true
    }
)
```

Figura 44 – Consulta ao banco de dados de log para identificar fraudador interno

A Figura 45 mostra o resultado do banco de dados, que possuía apenas uma aprovação de solicitação de emissão de certificado com probabilidade de fraude de 95%, que foi aprovada pelo usuário detentor do ID de valor 1. Essa informação pode ser investigada e levar a um fraudador interno, considerando as exceções em que a pontuação alta não implique em fraude.


```
{
    "_id" : ObjectId("59d7b439dace97125efad6b5"),
    "_class" : "br.ufsc.labsec.fraudclassifier.proxy.model.Log",
    "idEntrada" : ObjectId("59d7b42edace97125de7ac18"),
    "idUsuario" : NumberLong("1"),
    "aprovada" : true,
    "probabilidadeFraude" : 0.9545414106495486,
    "probabilidadeAutentica" : 0.045458589350568186
}
```


Figura 45 – Resultado da consulta ao banco de dados de log

5 CONCLUSÃO E TRABALHOS FUTUROS

Assinaturas e certificados digitais estão cada vez mais inseridos em atividades cotidianas nos sistemas eletrônicos para garantir a identidade das pessoas. O mercado de certificação tem uma clara tendência de crescimento, e os dados gerados sobre as emissões podem revelar irregularidades, que muitas vezes passam despercebidas pelos indivíduos. Já existe um processo de comunicação de irregularidades dentro da ICP-Brasil, mas este mostra-se demorado e mais suscetível a erros, já que considera a intuição dos indivíduos e não leva em conta o conhecimento incorporado nesses dados.

Com um estudo mais detalhado sobre a emissão de certificados digitais, foi possível identificar dois tipos de fraudadores: o fraudador interno, que foi corrompido e deixa a fraude acontecer, e o fraudador externo se passando por solicitante do certificado. Dentro deste contexto, foram executadas as etapas do processo de KDD, e identificados os dados envolvidos neste processo, e qual etapa é a mais adequada para executar um serviço de detecção de fraudes, neste caso, a etapa de validação presencial.

Como resultado do KDD, obteve-se um modelo de *Data Mining* para classificação de novas entradas, que informa ao agente de registro a probabilidade de ser fraude. Para este trabalho, foi optado por aprendizado de máquina supervisionado, utilizando modelos de classificação *eager*.

Foram definidos dois modelos para a integração com a ICP-Brasil, dos quais optou-se por implementar o modelo que faz uso de um proxy para comunicação segura, uma vez que este se mostrou de mais fácil implantação por parte da AR. Para tal, foram desenvolvidos os módulos do Serviço de Detecção de Fraudes e o Proxy, bem como manuais de instruções para a AC-Raiz e as ARs.

Comprovou-se a viabilidade da proposta através da integração com um sistema de AR fictício, onde todo o fluxo da proposta foi simulado. O resultado desta simulação foi a detecção das atividades de um possível fraudador externo, que obteve alta pontuação para fraude, e de um fraudador interno, que aceitou emitir um certificado para um possível solicitante fraudador.

Como trabalhos futuros, ficou definido: a validação dos dados relacionados à identidade do requerente oriundos de buscas pelo CPF do requerente em bases de dados confiáveis; a utilização de métodos reais para a obtenção dos valores de similaridades ao invés de valores

pseudoaleatórios; utilização de outras técnicas de *Data Mining* com aprendizado não supervisionado; a integração do serviço de detecção na ICP-Brasil juntamente com ACs e ARs, e avaliação dos modelos de *Data Mining* gerados a partir de dados reais.

REFERÊNCIAS

ADAMS, C.; LLOYD, S. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Boston, EUA: Addison-Wesley, 2003.

AHMADPANAH, S. H. What is git?! CreateSpace Independent Publishing Platform, 2015.

ANGELONI, M. T. Elementos intervenientes na tomada de decisão. *Ci. Inf.*, v. 32, p. 17–22, 2003.

APACHE. *Apache Maven Project*. 2017. <https://maven.apache.org/what-is-maven.html>. Acessado em 23/08/2017.

AZEVEDO, A. I. R. L. Kdd, semma and crisp-dm: a parallel overview. *IADS-DM,*, v. 1, p. 182–185, 2008.

BARSE, E. L.; KVARNSTROM, H.; JONSSON, E. Synthesizing test data for fraud detection systems. In: IEEE. *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. [S.l.], 2003. p. 384–394.

BHATLA, T. P.; PRABHU, V.; DUA, A. Understanding credit card frauds. *Cards Business Review2003-01*, 2003.

BOLTON, R. J.; HAND, D. J. et al. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, Imperial College London, UK, p. 235–255, 2001.

BRASIL. *Constituição: República Federativa do Brasil*. [S.l.]: DF: Senado Federal, 1988.

BRASIL. *Medida Provisória Nº 2.200-1, de 8 de agosto de 2001*. 2001. <http://www.planalto.gov.br/ccivil_03/mpv/antigas_2001/2200-2.htm>. Acessado em 30/03/2017.

BRASIL. *Decreto - Lei Nº 12.527, de 18 de novembro de 2011*. 2011. <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm>. Acessado em 28/03/2017.

BRASIL, P. *PF investiga fraudes no FGTS em Pernambuco e São Paulo*. 2017. <http://www.brasil.gov.br/defesa-e-seguranca/2016/10/pf-investiga-fraudes-no-fgts-em-pernambuco-e-sao-paulo>. Acessado em 22/03/2017.

BRASIL, P. C. *O mercado de certificação digital brasileiro deverá ultrapassar R$ 2 bilhões em 2020*. 2017. <http://portalcdbrasil.com.br/o-mercado-de-certificacao-digital-brasileiro-devera-ultrapassar-2-bilhoes-em-2020/>. Acessado em 22/03/2017.

BURGE, P.; SHAWE-TAYLOR, J. An unsupervised neural network approach to profiling the behavior of mobile phone users for use in fraud detection. *Journal of parallel and distributed computing*, Elsevier, v. 61, n. 7, p. 915–925, 2001.

CAHILL, M. et al. Detecting fraud in the real world [secção do livro]. *Handbook of massive data sets.-USA: Kluwer Academic Publishers*, v. 20, n. 0, p. 20, 2002.

CARNEIRO, N.; FIGUEIRA, G.; COSTA, M. A data mining based system for credit-card fraud detection in e-tail. *Decision Support Systems*, 2017.

CARUANA, R.; NICULESCU-MIZIL, A. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In: ACM. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2004. p. 69–78.

CERTISIGN. *Indicações e uso do Certificado Digital*. 2017. <https://www.certisign.com.br/certificado-digital/indicacao-uso>. Acessado em 22/03/2017.

CHAN, P. K. et al. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, IEEE, v. 14, n. 6, p. 67–74, 1999.

CHAPMAN, P. et al. *CRISP-DM 1.0 - Step-by-step data mining guide*. EUA: CRISP-DM Consortium, 2000.

CHEN, R.-C. et al. Detecting credit card fraud by using questionnaire-responded transaction model based on support vector machines. *Intelligent Data Engineering and Automated Learning–IDEAL 2004*, Springer, p. 800–806, 2004.

DAVENPORT, T. H.; PRUSAK, L. *Conhecimento empresarial: como as empresas gerenciam seu capital intelectual*. Rio de Janeiro, Brasil: Campus-Elsevier, 1998.

FAWCETT, T. In vivo spam filtering: a challenge problem for kdd. *ACM SIGKDD Explorations Newsletter*, ACM, v. 5, n. 2, p. 140–148, 2003.

FAWCETT, T.; PROVOST, F. Activity monitoring: Noticing interesting changes in behavior. In: ACM. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 1999. p. 53–62.

FOSTER, D. P.; STINE, R. A. Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association*, Taylor & Francis, v. 99, n. 466, p. 303–313, 2004.

GHOSH, S.; REILLY, D. L. Credit card fraud detection with a neural-network. In: IEEE. *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*. [S.l.], 1994. v. 3, p. 621–630.

GOMES, F.

Proposta de um modelo de transparência de dados e informações do processo de venda e emissão de certificados digitais para a governança do ITI — Universidade Federal de Santa Catarina, 2017.

HOUSLEY, R.; POLK, T. *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. Indiana, EUA: John Wiley & Sons, Inc., 2001.

ITI. *Certificação Digital ICP-Brasil*. 2017. <http://www.iti.gov.br/certificacao-digital>. Acessado em 04/04/2017.

ITI. *Institucional ITI*. 2017. <http://www.iti.gov.br/institucional>. Acessado em 03/04/2017.

ITI. *O que é ICP-Brasil*. 2017. <http://www.iti.gov.br/icp-brasil>. Acessado em 20/03/2017.

ITI. *Procedimentos para Identificação do Requerente e Comunicação de Irregularidades no processo de emissão de um Certificado Digital ICP-Brasil*. 2017. <http://www.iti.gov.br/images/repositorio/legislacao/documentos-principais/DOC-ICP-05.02_Versao_1.3_Procedimentos_de_Identificao_do_Requerente.pdf>. Acessado em 20/03/2017.

ITI. *Requisitos mínimos para as políticas de certificado na ICP-Brasil*. 2017. <http://iti.gov.br/images/repositorio/legislacao/documentos-principais/DOC-ICP-04—Versao-6.2.pdf>. Acessado em 20/03/2017.

KAHN, D. *The Codebreakers, The Story of Secret Writing*. New York, EUA: The Macmillan Company, 1967.

KIM, J.; ONG, A.; OVERILL, R. E. Design of an artificial immune system as a novel anomaly detector for combating financial fraud in the retail sector. In: IEEE. *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. [S.l.], 2003. v. 1, p. 405–412.

LEE, W.; XIANG, D. Information-theoretic measures for anomaly detection. In: IEEE. *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. [S.l.], 2001. p. 130–143.

LITTLE, B. B. et al. Collusion in the us crop insurance program: applied data mining. In: SIAM. *Proceedings of the 2002 SIAM International Conference on Data Mining*. [S.l.], 2002. p. 583–597.

LUGER, G. *Inteligência Artificial*. São Paulo, Brasil: Pearson Education do Brasil Ltda, 2013.

MAIMON, O.; ROKACH, L. *Data Mining and Knowledge Discovery Handbook*. Boston, EUA: Springer Science+Business Media, Inc., 2005.

MASSE, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. [S.l.]: "O'Reilly Media, Inc.", 2011.

MENEZES, A. J.; OORSCHOT, P. C. van; VANSTONE, S. A. *Handbook of Applied Cryptography*. Boca Raton, EUA: CRC Press, 1996.

MONGODB. *What is MongoDB*. 2017. <https://www.mongodb.com/what-is-mongodb>. Acessado em 23/08/2017.

ORACLE. *The History of Java Technology*. 2017. <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>. Acessado em 27/08/2017.

ORACLE. *Java Persistence API*. 2017. <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>. Acessado em 23/08/2017.

PAUTASSO, C. Restful web service composition with bpel for rest. *Data & Knowledge Engineering*, Elsevier, v. 68, n. 9, p. 851–866, 2009.

PEI, J.; HAN, J.; KAMBER, M. *Data mining: Concepts and techniques*. [S.l.]: Elsevier, 2012.

PHUA, C. et al. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

PRODANOV, C.; FREITAS, E. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico*. Rio Grande do Sul - Brasil: Feevale, 2013.

RITTER, T. *Ritter's Crypto Glossary and Dictionary of Technical Cryptography*. 2007. <http://ciphersbyritter.com/GLOSSARY.HTM>. Acessado em 23/03/2017.

SPRING. *Spring Boot Reference Guide*. 2017. <http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/getting-started-introducing-spring-boot>. Acessado em 23/08/2017.

STALLINGS, W. *Criptografia e segurança de redes: princípios e práticas*. [S.l.]: Pearson Prentice Hall, 2008.

VIAENE, S.; DERRIG, R. A.; DEDENE, G. A case study of applying boosting naive bayes to claim fraud diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 16, n. 5, p. 612–620, 2004.

WEISSMANN, H. L. *Vire o jogo com Spring Framework*. [S.l.]: Editora Casa do Código, 2014.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016.

YAMANISHI, K. et al. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, Springer Science & Business Media, v. 8, n. 3, p. 275, 2004.

ANEXO A – Manual do SDF

MANUAL PARA UTILIZAÇÃO DO SERVIÇO DE DETECÇÃO DE FRAUDE - AUTORIDADE CERTIFICADORA RAIZ

1 Introdução

O Serviço de Detecção de Fraudes para emissão de certificados digitais ICP-Brasil tem como objetivo gerar uma pontuação de possível fraude em novas solicitações de emissão de certificados. O serviço é acionado durante a etapa de validação presencial, onde o requerente apresenta-se pessoalmente com seus documentos. Os dados são enviados através de um web service para serem executados em um modelo de Data Mining, e retornar a pontuação. Este modelo de Data Mining também é mantido atualizado com as novas entradas classificadas.

Este manual tem o propósito de auxiliar a Autoridade Certificadora Raiz a implantar este modelo de detecção de fraudes em seu processo de detecção de fraudes. Aqui se encontram todas as informações necessárias para realizar com sucesso a implantação do serviço e como gerenciá-lo.

2 Público Alvo

As informações que constam neste manual são indicadas para analistas de sistemas ou desenvolvedores de software para Autoridade Certificadora Raiz da ICP-Brasil.

3 Pré-requisitos técnicos

- Conhecimento intermediário com linguagem de programação;
- Conhecimento de aplicações Web.

4 Procedimentos

4.1 Instalação do *web service Serviço de Detecção de Fraude (SDF)*

A AC-Raiz deve fazer o deploy do *web service* SDF em algum servidor que permita que a comunicação com os proxys que serão instalados no ambiente das Autoridades Registradoras . Para isso, é necessário instalar o MongoDB (preferencialmente na versão 3.2.16) e fazer as devidas configurações para segurança. Após isso, o arquivo de extensão JAR deve ser executado com o Java na versão 1.8.0.

Existem algumas configurações possíveis que podem ser feitas no *Proxy*. Estas podem ser feitas com o uso de parâmetros no comando de execução do JAR da seguinte forma:

```
$ java -jar fraud-classifier-VERSAO.jar [--prop.ex1=valor1]
```

As propriedades configuráveis desta forma são as seguintes:

Propriedade	**Tipo de dado**	**Descrição**	**Valor Padrão**
`server.port`	Porta	Porta em que o serviço executará	`8443`
`server.ssl.enabled`	Boolean	Habilita ou não o SSL	`true`
`server.ssl.key-store`	Caminho para arquivo (absoluto ou relativo)	Área de armazenamento de chaves usada para autenticar-se com o *fraud-classifier*	`(não definido)`
`server.ssl.key-store-password`	Texto	Senha da área de armazenamento de chaves	`(não definido)`
`server.ssl.key-store-type`	Tipo de área de armazenamento de chave (Ex: JKS ou PKCS12)	Tipo da área do armazenamento de chave usado na autenticação	`(não definido)`
`server.ssl.trust-store`	Caminho para arquivo (absoluto ou relativo)	Área de armazenamento de chaves em que o *fraud-classifier* permite autenticação	`(não definido)`
`server.ssl.trust-store-password`	Texto	Senha da área de armazenamento de chaves	`(não definido)`
`server.ssl.trust-store-type`	Tipo de área de armazenamento de chave (Ex: JKS ou PKCS12)	Tipo da área do armazenamento de chave usado na autenticação	`(não definido)`
`server.ssl.client-auth`	`want` | `need`	Caso o cliente precise autenticar-se ou não	`need`
`spring.data.mongodb.host`	IP	Endereço em que o *MongoDB* está executando	`localhost`
`spring.data.mongodb.port`	Porta	Porta em que o MongoDB está executando	`27017`
`spring.data.mongodb.database`	Texto	Nome do banco de dados usado pelo MongoDB	`fraudclassifier`
`spring.data.mongodb.username`	Texto	Nome de usuário para acesso ao MongoDB	`(não definido)`
`spring.data.mongodb.password`	Texto	Senha para acesso ao	`(não definido)`

		MongoDB	
`security.admin.cn`	Texto	Nome comum que será considerado para definir se um usuário é administrador	`admin.localhost`
`weka.classifier`	Classe JAVA	Implementação de classificador do Weka usada (deve ser uma implementação de `weka.classifiers.Classifier`)	`weka.classifiers.trees.J48`
`loggin.file`	Caminho para arquivo (absoluto ou relativo)	Arquivo em que o log da aplicação será armazenado	`../fraudclassifier.log`

4.2 Serviços do *web service* REST

Ação	Visualizar estatísticas do classificador
URL	`/evaluate?folds=:folds&arff=:gerarArff`
Método	GET
Parâmetros da URL (* campo opcional)	`* folds: integer (10)` `* gerarArff: boolean (false)`
Descrição dos parâmetros da URL	○ **`folds`:** número de partições feitas no conjunto de dados para efetuar a validação cruzada ○ **`gerarArff`:** se deve ser gerado um arquivo ARFF com as entradas classificadas do banco de dados na mesma pasta do programa em execução
Códigos de resposta	OK (200) | Bad Request (400)

ANEXO B – Manual do *web service* Proxy

MANUAL PARA UTILIZAÇÃO DO SERVIÇO DE DETECÇÃO DE FRAUDE - AUTORIDADE REGISTRADORA

1 Introdução

O Serviço de Detecção de Fraudes para emissão de certificados digitais ICP-Brasil tem como objetivo gerar uma pontuação de possível fraude em novas solicitações de emissão de certificados. O serviço é acionado durante a etapa de validação presencial, onde o requerente apresenta-se pessoalmente com seus documentos. Os dados são enviados através de um web service para serem executados em um modelo de Data Mining, e retornar a pontuação. Este modelo de Data Mining também é mantido atualizado com as novas entradas classificadas.

Este manual tem o propósito de auxiliar as Autoridades Registradoras a implantarem este modelo de detecção de fraudes em seu processo de emissão. Aqui se encontram todas as informações necessárias para realizar com sucesso a integração do serviço ao sistema utilizado para gerenciar a emissão do certificado digital.

2 Público Alvo

As informações que constam neste manual são indicadas para analistas de sistemas ou desenvolvedores de software para os sistemas de Autoridades Registradoras e Autoridades Certificadoras da ICP-Brasil.

3 Pré-requisitos técnicos

- Conhecimento intermediário com linguagem de programação;
- Conhecimento de aplicações Web.

4 Procedimentos

4.1 Instalação do *web service Proxy*

A Autoridade Registradora deve fazer o deploy do *web service Proxy* em um servidor próprio, no alcance do sistema que gerencia a emissão do Certificado Digital. Para isso, é necessário instalar o MongoDB (preferencialmente na versão 3.2.16) e fazer as devidas configurações para segurança. Após isso, o arquivo de extensão JAR deve ser executado com o Java na versão 1.8.0.

Existem algumas configurações possíveis que podem ser feitas no *Proxy*. Estas podem ser feitas com o uso de parâmetros no comando de execução do JAR da seguinte forma:

```
$ java -jar fraud-classifier-proxy-VERSAO.jar [--prop.ex1=valor1]
```

As propriedades configuráveis desta forma são as seguintes:

Propriedade	**Tipo de dado**	**Descrição**	**Valor Padrão**
`server.port`	Porta	Porta em que o	`8442`

		serviço executará	
`server.ssl.enabled`	Boolean	Habilita ou não o SSL	`true`
`server.ssl.key-store`	Caminho para arquivo (absoluto ou relativo)	Área de armazenamento de chaves usada para autenticar-se com o *fraud-classifier*	`(não definido)`
`server.ssl.key-store-password`	Texto	Senha da área de armazenamento de chaves	`(não definido)`
`server.ssl.key-store-type`	Tipo de área de armazenamento de chave (Ex: JKS ou PKCS12)	Tipo da área do armazenamento de chave usado na autenticação	`(não definido)`
`fraud-classifier.host`	URL	Endereço em que o *fraud-classifier* está executando	`https://localhost:8443`
`spring.data.mongodb.host`	IP	Endereço em que o *MongoDB* está executando	`localhost`
`spring.data.mongodb.port`	Porta	Porta em que o MongoDB está executando	`27017`
`spring.data.mongodb.database`	Texto	Nome do banco de dados usado pelo MongoDB	`log`
`spring.data.mongodb.username`	Texto	Nome de usuário para acesso ao MongoDB	`(não definido)`
`spring.data.mongodb.password`	Texto	Senha para acesso ao MongoDB	`(não definido)`

4.2 Serviços do *web service* REST

Ação	Classificar uma entrada baseado em dados na solicitação de emissão e validação presencial
URL	`/classify`
Método	POST
Parâmetros da requisição	`{` `  foto: byte[],`

	imagemDocumento: byte[], assinatura: byte[], assinaturaDocumento: byte[], digital: byte[], digitalDocumento: byte[], tipo: string, cidadeSolicitante: string, ufSolicitante: string, razaoSocialAc: string, razaoSocialAr: string, dataNascimento: date, dataExpedicaoRg: date, dataExpedicaoCnh: date, dataExpedicaoCne: date, dataExpedicaoPassaporte: date, dataExpedicaoCnpj: date, dataExpedicaoCCProfissional: date, dataExpedicaoCertidao: date, dataExpedicaoCtps: date }
Descrição dos parâmetros da requisição	○ **foto:** Foto do requerente capturada na validação presencial ○ **imagemDocumento:** Imagem do requerente presente no documento de identificação ○ **assinatura:** Assinatura do requerente colhida na validação presencial ○ **assinaturaDocumento:** Assinatura do requerente presente no documento de identificação ○ **digital:** Digital do requerente colhida na validação presencial ○ **digitalDocumento:** Digital do requerente presente no documento de identificação ○ **tipo:** Política do certificado (Ex: A1) ○ **cidadeSolicitante:** Cidade de residência do requerente ○ **ufSolicitante:** UF de residência do requerente ○ **razaoSocialAc:** Razão social da AC ○ **razaoSocialAr:** Razão social da AR ○ **dataNascimento:** Data de nascimento do requerente ○ *** dataExpedicaoRg:** Data de emissão do Registro Geral do requerente ○ *** dataExpedicaoCnh:** Data de emissão da Carteira Nacional de Habilitação do requerente ○ *** dataExpedicaoCne:** Data de emissão da Carteira Nacional de Estrangeiro do requerente ○ *** dataExpedicaoPassaporte:** Data de emissão do Passaporte Brasileiro do requerente

	○ **`* dataExpedicaoCCProfissional:`** Data de emissão da Carteira do Conselho Profissional do requerente ○ **`* dataExpedicaoCtps:`** Data de emissão da Carteira de Trabalho do requerente ○ **`** dataExpedicaoCnpj:`** Data de emissão do CNPJ do requerente ○ **`** dataExpedicaoCertidao:`** Data de emissão da Certidão de Nascimento do requerente *** Um dos campos obrigatório** **** Campo opcional**
Códigos de resposta	OK (200) | Bad Request (400)

Corpo da Resposta	`{`
	`  id: string,`
	`  probabilidadeFraude: double,`
	`  probabilidadeAutentica: double`
	`}`

Descrição do corpo da resposta	○ **`id:`** Identificador no MongoDB da entrada classificada ○ **`probabilidadeFraude:`** Valor da probabilidade da entrada tratar-se de uma fraude num intervalo de 0 a 1 ○ **`probabilidadeAutentica:`** Valor da probabilidade da entrada tratar-se de uma solicitação autêntica num intervalo de 0 a 1

Ação	Definir a classe de uma entrada já existente
URL	`/classify/:id?f=:fraude`
Método	PUT
Parâmetros da URL	`id: string` `fraude: boolean`
Descrição dos parâmetros da URL	○ **`id:`** Identificador da entrada que se deseja definir a classe ○ **`fraude:`** Classe a ser atribuída à entrada
Parâmetros da requisição	`idUsuario: long`
Descrição dos parâmetros da requisição	○ **`idUsuario:`** Identificador do usuário no sistema de gerência de solicitações de emissão de certificados que realizou a ação
Códigos de resposta	OK (200) | Bad Request (400)
Descrição do corpo da resposta	○ **`id:`** Identificador no MongoDB da entrada classificada

	○ **`probabilidadeFraude:`** Valor da probabilidade da entrada tratar-se de uma fraude num intervalo de 0 a 1 ○ **`probabilidadeAutentica:`** Valor da probabilidade da entrada tratar-se de uma solicitação autêntica num intervalo de 0 a 1

ANEXO C – Código fonte dos projetos

1 fraud-classifier-commons

1.1 SolicitacaoEmissao.java

```
package br.ufsc.labsec.fraudclassifier.model.commons;

import java.util.Calendar;

public class SolicitacaoEmissao {

  private Calendar dataNascimento;

  private byte[] foto;
  private byte[] imagemDocumento;

  private byte[] assinatura;
  private byte[] assinaturaDocumento;

  private byte[] digital;
  private byte[] digitalDocumento;

  private Calendar dataExpedicaoRg;
  private Calendar dataExpedicaoCnh;
  private Calendar dataExpedicaoCne;
  private Calendar dataExpedicaoPassaporte;
  private Calendar dataExpedicaoCtps;
  private Calendar dataExpedicaoCCProfissional;

  private Calendar dataExpedicaoCnpj;
  private Calendar dataExpedicaoCertidao;

  private String tipo;
  private String cidadeSolicitante;
  private String ufSolicitante;
  private String razaoSocialAc;
  private String razaoSocialAr;

  public Calendar getDataNascimento() {
    return dataNascimento;
  }

  public void setDataNascimento(final Calendar dataNascimento) {
    this.dataNascimento = dataNascimento;
  }

  public byte[] getFoto() {
    return foto;
  }

  public void setFoto(final byte[] foto) {
    this.foto = foto;
  }

  public byte[] getImagemDocumento() {
    return imagemDocumento;
  }
```

```
public void setImagemDocumento(final byte[] imagemDocumento) {
  this.imagemDocumento = imagemDocumento;
}

public byte[] getAssinatura() {
  return assinatura;
}

public void setAssinatura(final byte[] assinatura) {
  this.assinatura = assinatura;
}

public byte[] getAssinaturaDocumento() {
  return assinaturaDocumento;
}

public void setAssinaturaDocumento(final byte[] assinaturaDocumento) {
  this.assinaturaDocumento = assinaturaDocumento;
}

public byte[] getDigital() {
  return digital;
}

public void setDigital(final byte[] digital) {
  this.digital = digital;
}

public byte[] getDigitalDocumento() {
  return digitalDocumento;
}

public void setDigitalDocumento(final byte[] digitalDocumento) {
  this.digitalDocumento = digitalDocumento;
}

public Calendar getDataExpedicaoRg() {
  return dataExpedicaoRg;
}

public void setDataExpedicaoRg(final Calendar dataExpedicaoRg) {
  this.dataExpedicaoRg = dataExpedicaoRg;
}

public Calendar getDataExpedicaoCnh() {
  return dataExpedicaoCnh;
}

public void setDataExpedicaoCnh(final Calendar dataExpedicaoCnh) {
  this.dataExpedicaoCnh = dataExpedicaoCnh;
}

public Calendar getDataExpedicaoCne() {
  return dataExpedicaoCne;
}
```

```
public void setDataExpedicaoCne(final Calendar dataExpedicaoCne) {
  this.dataExpedicaoCne = dataExpedicaoCne;
}

public Calendar getDataExpedicaoPassaporte() {
  return dataExpedicaoPassaporte;
}

public void setDataExpedicaoPassaporte(final Calendar dataExpedicaoPassaporte
    ) {
  this.dataExpedicaoPassaporte = dataExpedicaoPassaporte;
}

public Calendar getDataExpedicaoCtps() {
  return dataExpedicaoCtps;
}

public void setDataExpedicaoCtps(final Calendar dataExpedicaoCtps) {
  this.dataExpedicaoCtps = dataExpedicaoCtps;
}

public Calendar getDataExpedicaoCCProfissional() {
  return dataExpedicaoCCProfissional;
}

public void setDataExpedicaoCCProfissional(final Calendar
    dataExpedicaoCCProfissional) {
  this.dataExpedicaoCCProfissional = dataExpedicaoCCProfissional;
}

public Calendar getDataExpedicaoCnpj() {
  return dataExpedicaoCnpj;
}

public void setDataExpedicaoCnpj(final Calendar dataExpedicaoCnpj) {
  this.dataExpedicaoCnpj = dataExpedicaoCnpj;
}

public Calendar getDataExpedicaoCertidao() {
  return dataExpedicaoCertidao;
}

public void setDataExpedicaoCertidao(final Calendar dataExpedicaoCertidao) {
  this.dataExpedicaoCertidao = dataExpedicaoCertidao;
}

public String getTipo() {
  return tipo;
}

public void setTipo(final String tipo) {
  this.tipo = tipo;
}

public String getCidadeSolicitante() {
  return cidadeSolicitante;
```

```
  }

  public void setCidadeSolicitante(final String cidadeSolicitante) {
    this.cidadeSolicitante = cidadeSolicitante;
  }

  public String getUfSolicitante() {
    return ufSolicitante;
  }

  public void setUfSolicitante(final String ufSolicitante) {
    this.ufSolicitante = ufSolicitante;
  }

  public String getRazaoSocialAc() {
    return razaoSocialAc;
  }

  public void setRazaoSocialAc(final String razaoSocialAc) {
    this.razaoSocialAc = razaoSocialAc;
  }

  public String getRazaoSocialAr() {
    return razaoSocialAr;
  }

  public void setRazaoSocialAr(final String razaoSocialAr) {
    this.razaoSocialAr = razaoSocialAr;
  }

}
```

1.2 Classificacao.java

```
package br.ufsc.labsec.fraudclassifier.model.commons;

import org.apache.commons.lang3.builder.ToStringBuilder;

public class Classificacao {

  private String id;
  private Double probabilidadeFraude;
  private Double probabilidadeAutentica;

  @Override
  public String toString() {
    return ToStringBuilder.reflectionToString(this);
  }

  public String getId() {
    return id;
  }

  public void setId(final String id) {
    this.id = id;
  }
```

```
  public Double getProbabilidadeFraude() {
    return probabilidadeFraude;
  }

  public void setProbabilidadeFraude(final Double probabilidadeFraude) {
    this.probabilidadeFraude = probabilidadeFraude;
  }

  public Double getProbabilidadeAutentica() {
    return probabilidadeAutentica;
  }

  public void setProbabilidadeAutentica(final Double probabilidadeAutentica) {
    this.probabilidadeAutentica = probabilidadeAutentica;
  }

}
```

2 fraud-classifier

2.1 App.java

```
package br.ufsc.labsec.fraudclassifier;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.support.SpringBootServletInitializer;
import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
public class App extends SpringBootServletInitializer {

  public static void main(final String[] args) {
    SpringApplication.run(App.class, args);
  }

}
```

2.2 SecurityConfig.java

```
package br.ufsc.labsec.fraudclassifier.security;

import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.security.config.annotation.method.configuration.
    EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity
    ;
import org.springframework.security.config.annotation.web.configuration.
    EnableWebSecurity;
```

```
import org.springframework.security.config.annotation.web.configuration.
    WebSecurityConfigurerAdapter;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;

@SpringBootApplication
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
@PropertySource("classpath:application.properties")
public class SecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  private Environment env;

  @Override
  protected void configure(final HttpSecurity http) throws Exception {
    // @formatter:off
    http
      .authorizeRequests().anyRequest().authenticated()
      .and().x509().subjectPrincipalRegex("CN=(.*?)(?:,|$)").userDetailsService
          (userDetailsService())
      .and().csrf().disable();
    // @formatter:on
  }

  @Override
  @Bean
  public UserDetailsService userDetailsService() {
    return username -> {
      if(StringUtils.isNotBlank(username)) {
        if(StringUtils.equals(username, env.getProperty("security.admin.cn")))
            {
          return new User(username, "", AuthorityUtils.createAuthorityList("
              ROLE_ADMIN"));
        }
        return new User(username, "", AuthorityUtils.createAuthorityList("
            ROLE_USER"));
      }
      return null;
    };
  }

}
```

2.3 ClassifierConfig.java

```
package br.ufsc.labsec.fraudclassifier.classification.config;

import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
```

```
import weka.classifiers.Classifier;
import weka.classifiers.trees.J48;

@SpringBootConfiguration
@PropertySource({ "classpath:application.properties" })
public class ClassifierConfig {

  private static final String DEFAULT_CLASSIFIER = J48.class.getName();

  @Autowired
  private Environment env;

  @Bean
  public Classifier classifier() {
    final String className = StringUtils.defaultIfBlank(env.getProperty("weka.
        classifier"), DEFAULT_CLASSIFIER);
    try {
      return (Classifier) Class.forName(className).newInstance();
    } catch(final ClassNotFoundException | InstantiationException |
        IllegalAccessException e) {
      return new J48();
    }
  }

}
```

2.4 Entrada.java

```
package br.ufsc.labsec.fraudclassifier.model;

import java.util.Date;

import org.apache.commons.lang3.builder.ToStringBuilder;
import org.bson.types.ObjectId;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import br.ufsc.labsec.fraudclassifier.classification.annotation.
    GeneratesFraudProbability;
import br.ufsc.labsec.fraudclassifier.classification.annotation.Ignore;
import br.ufsc.labsec.fraudclassifier.classification.annotation.Label;

@Document(collection = "entradas")
public class Entrada {

  @Id
  private ObjectId id;

  private Date dataNascimento;

  private Double similaridadeFotos;
  private Double similaridadeAssinaturas;
  private Double similaridadeDigitais;

  private String tipo;
```

```
@GeneratesFraudProbability
private String cidadeSolicitante;
@GeneratesFraudProbability
private String ufSolicitante;
@GeneratesFraudProbability
private String razaoSocialAc;
@GeneratesFraudProbability
private String razaoSocialAr;

private Boolean dataExpedicaoDiaUtilRg;
private Boolean dataExpedicaoDiaUtilCnh;
private Boolean dataExpedicaoDiaUtilCne;
private Boolean dataExpedicaoDiaUtilPassaporte;
private Boolean dataExpedicaoDiaUtilCtps;
private Boolean dataExpedicaoDiaUtilCCProfissional;

private Boolean dataExpedicaoDiaUtilCnpj;
private Boolean dataExpedicaoDiaUtilCertidao;

@Label
private Boolean fraude;

@Ignore
private Double probabilidadeFraude;
@Ignore
private Double probabilidadeAutentica;

@Override
public String toString() {
  return ToStringBuilder.reflectionToString(this);
}

public ObjectId getId() {
  return id;
}

public void setId(final ObjectId id) {
  this.id = id;
}

public Date getDataNascimento() {
  return dataNascimento;
}

public void setDataNascimento(final Date dataNascimento) {
  this.dataNascimento = dataNascimento;
}

public Double getSimilaridadeFotos() {
  return similaridadeFotos;
}

public void setSimilaridadeFotos(final Double similaridadeFotos) {
  this.similaridadeFotos = similaridadeFotos;
}
```

```
public Double getSimilaridadeAssinaturas() {
  return similaridadeAssinaturas;
}

public void setSimilaridadeAssinaturas(final Double similaridadeAssinaturas)
   {
  this.similaridadeAssinaturas = similaridadeAssinaturas;
}

public Double getSimilaridadeDigitais() {
  return similaridadeDigitais;
}

public void setSimilaridadeDigitais(final Double similaridadeDigitais) {
  this.similaridadeDigitais = similaridadeDigitais;
}

public String getTipo() {
  return tipo;
}

public void setTipo(final String tipo) {
  this.tipo = tipo;
}

public String getCidadeSolicitante() {
  return cidadeSolicitante;
}

public void setCidadeSolicitante(final String cidadeSolicitante) {
  this.cidadeSolicitante = cidadeSolicitante;
}

public String getUfSolicitante() {
  return ufSolicitante;
}

public void setUfSolicitante(final String ufSolicitante) {
  this.ufSolicitante = ufSolicitante;
}

public String getRazaoSocialAc() {
  return razaoSocialAc;
}

public void setRazaoSocialAc(final String razaoSocialAc) {
  this.razaoSocialAc = razaoSocialAc;
}

public String getRazaoSocialAr() {
  return razaoSocialAr;
}

public void setRazaoSocialAr(final String razaoSocialAr) {
  this.razaoSocialAr = razaoSocialAr;
}
```

```
public Boolean isDataExpedicaoDiaUtilRg() {
  return dataExpedicaoDiaUtilRg;
}

public void setDataExpedicaoDiaUtilRg(final Boolean dataExpedicaoDiaUtilRg) {
  this.dataExpedicaoDiaUtilRg = dataExpedicaoDiaUtilRg;
}

public Boolean isDataExpedicaoDiaUtilCnh() {
  return dataExpedicaoDiaUtilCnh;
}

public void setDataExpedicaoDiaUtilCnh(final Boolean dataExpedicaoDiaUtilCnh)
    {
  this.dataExpedicaoDiaUtilCnh = dataExpedicaoDiaUtilCnh;
}

public Boolean isDataExpedicaoDiaUtilCne() {
  return dataExpedicaoDiaUtilCne;
}

public void setDataExpedicaoDiaUtilCne(final Boolean dataExpedicaoDiaUtilCne)
    {
  this.dataExpedicaoDiaUtilCne = dataExpedicaoDiaUtilCne;
}

public Boolean isDataExpedicaoDiaUtilPassaporte() {
  return dataExpedicaoDiaUtilPassaporte;
}

public void setDataExpedicaoDiaUtilPassaporte(final Boolean
   dataExpedicaoDiaUtilPassaporte) {
  this.dataExpedicaoDiaUtilPassaporte = dataExpedicaoDiaUtilPassaporte;
}

public Boolean isDataExpedicaoDiaUtilCtps() {
  return dataExpedicaoDiaUtilCtps;
}

public void setDataExpedicaoDiaUtilCtps(final boolean
   dataExpedicaoDiaUtilCtps) {
  this.dataExpedicaoDiaUtilCtps = dataExpedicaoDiaUtilCtps;
}

public Boolean isDataExpedicaoDiaUtilCCProfissional() {
  return dataExpedicaoDiaUtilCCProfissional;
}

public void setDataExpedicaoDiaUtilCCProfissional(final boolean
   dataExpedicaoDiaUtilCCProfissional) {
  this.dataExpedicaoDiaUtilCCProfissional =
     dataExpedicaoDiaUtilCCProfissional;
}

public Boolean isDataExpedicaoDiaUtilCnpj() {
```

```
    return dataExpedicaoDiaUtilCnpj;
  }

  public void setDataExpedicaoDiaUtilCnpj(final boolean
      dataExpedicaoDiaUtilCnpj) {
    this.dataExpedicaoDiaUtilCnpj = dataExpedicaoDiaUtilCnpj;
  }

  public Boolean isDataExpedicaoDiaUtilCertidao() {
    return dataExpedicaoDiaUtilCertidao;
  }

  public void setDataExpedicaoDiaUtilCertidao(final boolean
      dataExpedicaoDiaUtilCertidao) {
    this.dataExpedicaoDiaUtilCertidao = dataExpedicaoDiaUtilCertidao;
  }

  public Boolean isFraude() {
    return fraude;
  }

  public void setFraude(final boolean fraude) {
    this.fraude = fraude;
  }

  public Double getProbabilidadeFraude() {
    return probabilidadeFraude;
  }

  public void setProbabilidadeFraude(final Double probabilidadeFraude) {
    this.probabilidadeFraude = probabilidadeFraude;
  }

  public Double getProbabilidadeAutentica() {
    return probabilidadeAutentica;
  }

  public void setProbabilidadeAutentica(final Double probabilidadeAutentica) {
    this.probabilidadeAutentica = probabilidadeAutentica;
  }

}
```

2.5 EntradaDAO.java

```
package br.ufsc.labsec.fraudclassifier.persistence.dao;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.persistence.repository.EntradaRepository;
```

```
@Component
public class EntradaDAO {

  @Autowired
  private MongoTemplate mongoTemplate;

  @Autowired
  private EntradaRepository repository;

  private final Map<String, Map<Object, Double>> probabilities = new HashMap
     <>();

  public void refresh() {
    probabilities.clear();
  }

  @SuppressWarnings("unchecked")
  public List<String> findFieldDomain(final String field) {
    return mongoTemplate.getCollection("entradas").distinct(field);
  }

  public Double getFraudProbabilityForFieldAndValue(final String field, final
     Object value) {
    if(!probabilities.containsKey(field)) {
      probabilities.put(field, new HashMap<>());
    }
    final Map<Object, Double> probabilitiesForField = probabilities.get(field);
    Double prob = probabilitiesForField.get(value);
    if(prob == null) {
      prob = (double) repository.countFraudByFieldAndValue(field, value) /
         repository.countClassifiedByFieldAndValue(field, value);
      probabilitiesForField.put(value, prob);
    }
    return prob;
  }

}
```

2.6 EntradaRepository.java

```
package br.ufsc.labsec.fraudclassifier.persistence.repository;

import java.util.List;

import org.bson.types.ObjectId;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import br.ufsc.labsec.fraudclassifier.model.Entrada;

public interface EntradaRepository extends MongoRepository<Entrada, ObjectId> {

  @Query("{fraude: {$exists: true}}")
  List<Entrada> findClassified();
```

```
  @Query(value = "{?0:␣?1,␣fraude:␣{␣$exists:␣true}}", count = true)
  Long countClassifiedByFieldAndValue(String field, Object value);

  @Query(value = "{?0:␣?1,␣fraude:␣true}", count = true)
  Long countFraudByFieldAndValue(String field, Object value);

}
```

2.7 EntradaService.java

```
package br.ufsc.labsec.fraudclassifier.service;

import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import br.ufsc.labsec.fraudclassifier.classification.EntradaClassifier;
import br.ufsc.labsec.fraudclassifier.classification.comparation.
    BiometricSignatureComparator;
import br.ufsc.labsec.fraudclassifier.classification.comparation.
    BusinessDayVerifier;
import br.ufsc.labsec.fraudclassifier.classification.comparation.
    FacesComparator;
import br.ufsc.labsec.fraudclassifier.classification.comparation.
    WrittenSignatureComparator;
import br.ufsc.labsec.fraudclassifier.model.Entrada;
import br.ufsc.labsec.fraudclassifier.model.commons.Classificacao;
import br.ufsc.labsec.fraudclassifier.model.commons.SolicitacaoEmissao;
import br.ufsc.labsec.fraudclassifier.persistence.repository.EntradaRepository;

@Service
public class EntradaService {

  private static final Logger log = LoggerFactory.getLogger(EntradaService.
      class);

  @Autowired
  private EntradaRepository repository;

  @Autowired
  private FacesComparator facesComparator;

  @Autowired
  private WrittenSignatureComparator writtenSignatureComparator;

  @Autowired
  private BiometricSignatureComparator biometricSignatureComparator;

  @Autowired
  private BusinessDayVerifier businessDayVerifier;

  @Autowired
  private EntradaClassifier entradaClassifier;
```

```
@Transactional
public Entrada save(final SolicitacaoEmissao solicitacao) {
  final Entrada entrada = new Entrada();

  entrada.setDataNascimento(solicitacao.getDataNascimento().getTime());

  if(solicitacao.getDataExpedicaoRg() != null) {
    entrada.setDataExpedicaoDiaUtilRg(businessDayVerifier.isBusinessDay(
        solicitacao.getDataExpedicaoRg()));
  }
  if(solicitacao.getDataExpedicaoCnh() != null) {
    entrada.setDataExpedicaoDiaUtilCnh(businessDayVerifier.isBusinessDay(
        solicitacao.getDataExpedicaoCnh()));
  }
  if(solicitacao.getDataExpedicaoCne() != null) {
    entrada.setDataExpedicaoDiaUtilCne(businessDayVerifier.isBusinessDay(
        solicitacao.getDataExpedicaoCne()));
  }
  if(solicitacao.getDataExpedicaoPassaporte() != null) {
    entrada.setDataExpedicaoDiaUtilPassaporte(businessDayVerifier.
        isBusinessDay(solicitacao.getDataExpedicaoPassaporte()));
  }
  if(solicitacao.getDataExpedicaoCtps() != null) {
    entrada.setDataExpedicaoDiaUtilCtps(businessDayVerifier.isBusinessDay(
        solicitacao.getDataExpedicaoCtps()));
  }
  if(solicitacao.getDataExpedicaoCCProfissional() != null) {
    entrada.setDataExpedicaoDiaUtilCCProfissional(businessDayVerifier.
        isBusinessDay(solicitacao.getDataExpedicaoCCProfissional()));
  }

  if(solicitacao.getDataExpedicaoCnpj() != null) {
    entrada.setDataExpedicaoDiaUtilCnpj(businessDayVerifier.isBusinessDay(
        solicitacao.getDataExpedicaoCnpj()));
  }
  if(solicitacao.getDataExpedicaoCertidao() != null) {
    entrada.setDataExpedicaoDiaUtilCertidao(businessDayVerifier.isBusinessDay
        (solicitacao.getDataExpedicaoCertidao()));
  }

  entrada.setSimilaridadeFotos(facesComparator.compare(solicitacao.getFoto(),
      solicitacao.getImagemDocumento()));
  entrada.setSimilaridadeAssinaturas(writtenSignatureComparator.compare(
     solicitacao.getAssinatura(), solicitacao.getAssinaturaDocumento()));
  entrada.setSimilaridadeDigitais(biometricSignatureComparator.compare(
     solicitacao.getDigital(), solicitacao.getDigitalDocumento()));

  entrada.setTipo(solicitacao.getTipo());

  entrada.setCidadeSolicitante(StringUtils.stripAccents(solicitacao.
     getCidadeSolicitante()));
  entrada.setUfSolicitante(StringUtils.stripAccents(solicitacao.
     getUfSolicitante()));
  entrada.setRazaoSocialAc(StringUtils.stripAccents(solicitacao.
     getRazaoSocialAc()));
```

```
    entrada.setRazaoSocialAr(StringUtils.stripAccents(solicitacao.
        getRazaoSocialAr()));

    repository.save(entrada);

    log.info("Persisted␣Entry␣[{}]", entrada);

    return entrada;
  }

  @Transactional
  public Entrada setFraude(final Entrada entrada, final Boolean fraude) {
    entrada.setFraude(fraude);
    repository.save(entrada);

    log.info("Updated␣Entry␣[{}]:␣set␣fraude={}", entrada.getId(), fraude);

    return entrada;
  }

  public Classificacao classify(final Entrada entrada) throws Exception {
    Classificacao classificacao;

    if(entrada.getProbabilidadeFraude() != null && entrada.
        getProbabilidadeAutentica() != null) {
      classificacao = new Classificacao();
      classificacao.setId(entrada.getId().toString());
      classificacao.setProbabilidadeFraude(entrada.getProbabilidadeFraude());
      classificacao.setProbabilidadeAutentica(entrada.getProbabilidadeAutentica
          ());
    } else {
      classificacao = entradaClassifier.classify(entrada);

      entrada.setProbabilidadeFraude(classificacao.getProbabilidadeFraude());
      entrada.setProbabilidadeAutentica(classificacao.getProbabilidadeAutentica
          ());
      repository.save(entrada);

      log.info("Classified␣Entry␣[{}]:␣fraude=[true:{}␣/␣false:{}]",
        entrada.getId(),
        classificacao.getProbabilidadeFraude(),
        classificacao.getProbabilidadeAutentica());
    }

    return classificacao;
  }

}
```

2.8 EntradaClassifier.java

```
package br.ufsc.labsec.fraudclassifier.classification;

import java.io.File;
import java.lang.reflect.Field;
import java.util.ArrayList;
```

```
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.Objects;
import java.util.Random;
import java.util.stream.Collectors;

import javax.annotation.PostConstruct;

import org.apache.commons.lang3.StringUtils;
import org.apache.commons.lang3.reflect.FieldUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.classification.annotation.
   GeneratesFraudProbability;
import br.ufsc.labsec.fraudclassifier.classification.annotation.Ignore;
import br.ufsc.labsec.fraudclassifier.classification.annotation.Label;
import br.ufsc.labsec.fraudclassifier.model.Entrada;
import br.ufsc.labsec.fraudclassifier.model.commons.Classificacao;
import br.ufsc.labsec.fraudclassifier.persistence.dao.EntradaDAO;
import br.ufsc.labsec.fraudclassifier.persistence.repository.EntradaRepository;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.core.Attribute;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ArffSaver;

@Component
public class EntradaClassifier {

  private static final Logger log = LoggerFactory.getLogger(EntradaClassifier.
     class);

  @Autowired
  private Classifier classifier;

  @Autowired
  private EntradaRepository repository;

  @Autowired
  private EntradaDAO dao;

  private List<Attribute> attributes;
  private Instances trainData;

  @SuppressWarnings({ "rawtypes", "unchecked" })
  @PostConstruct
  @Scheduled(cron = "0␣0␣1␣*␣*␣*")
  public void init() throws Exception {
    attributes = FieldUtils.getAllFieldsList(Entrada.class).stream()
```

```
    .filter(f -> !f.isAnnotationPresent(Ignore.class))
    .sorted((f1, f2) -> f1.isAnnotationPresent(Label.class) ? 1 : f2.
       isAnnotationPresent(Label.class) ? -1 : 0)
    .map(this::fieldToAttribute)
    .filter(Objects::nonNull)
    .collect(Collectors.toList());

  trainData = new Instances("train", (ArrayList) attributes, attributes.size
     ());
  trainData.setClassIndex(attributes.size() - 1);

  dao.refresh();

  final List<Entrada> train = repository.findClassified();
  for(final Entrada e : train) {
    try {
      trainData.add(toInstance(e));
    } catch(final IllegalAccessException e1) {}
  }

  classifier.buildClassifier(trainData);

  log.info("New␣classifier␣built:\n{}", classifier);
}

private Attribute fieldToAttribute(final Field field) {
  final String fieldName = field.getName();
  if(field.isAnnotationPresent(GeneratesFraudProbability.class)) {
    return new AbstractAtribute(fieldName);
  } else {
    if(Double.class.equals(field.getType()) || Date.class.equals(field.
       getType())) {
      return new Attribute(fieldName);
    }
    if(Boolean.class.equals(field.getType())) {
      return new Attribute(fieldName, Arrays.asList("true", "false"));
    }
    if(String.class.equals(field.getType())) {
      return new Attribute(
        fieldName,
        dao.findFieldDomain(fieldName).stream()
          .map(StringUtils::stripAccents)
          .collect(Collectors.toList()));
    }
  }
  return null;
}

private Instance toInstance(final Entrada entrada) throws
    IllegalAccessException {
  final Instance i = new DenseInstance(attributes.size());
  for(final Attribute attribute : attributes) {
    final String fieldName = attribute.name();
    final Object value = FieldUtils.readDeclaredField(entrada, fieldName,
       true);
    if(attribute instanceof AbstractAtribute) {
```

```
      i.setValue(attribute, dao.getFraudProbabilityForFieldAndValue(fieldName
          , value));
    } else {
      if(value instanceof Double) {
        i.setValue(attribute, (double) value);
      } else if(value instanceof String) {
        i.setValue(attribute, StringUtils.stripAccents((String) value));
      } else if(value instanceof Boolean) {
        i.setValue(attribute, value.toString());
      } else if(value instanceof Date) {
        i.setValue(attribute, ((Date) value).getTime());
      }
    }
  }
  return i;
}

public Classificacao classify(final Entrada entrada) throws Exception {
  final Instance instance = toInstance(entrada);
  instance.setDataset(trainData);

  final double[] dist = classifier.distributionForInstance(instance);

  final Classificacao classificacao = new Classificacao();
  classificacao.setId(entrada.getId().toString());
  classificacao.setProbabilidadeFraude(dist[0]);
  classificacao.setProbabilidadeAutentica(dist[1]);
  return classificacao;
}

public String evaluateModel(final int folds, final boolean generateArf)
    throws Exception {
  if(trainData.isEmpty()) {
    final List<Entrada> train = repository.findClassified();
    for(final Entrada e : train) {
      try {
        trainData.add(toInstance(e));
      } catch(final IllegalAccessException e1) {}
    }
  }

  if(generateArf) {
    final ArffSaver saver = new ArffSaver();
    saver.setInstances(trainData);
    saver.setFile(new File("entradas.arff"));
    saver.writeBatch();
  }

  final Evaluation evaluation = new Evaluation(trainData);
  evaluation.crossValidateModel(classifier, trainData, folds, new Random());
  return new StringBuilder()
    .append(evaluation.toSummaryString())
    .append("<hr>")
    .append(classifier)
    .toString();
}
```

```
}
```

2.9 EntradaController.java

```
package br.ufsc.labsec.fraudclassifier.endpoint;

import java.util.Arrays;
import java.util.Calendar;
import java.util.List;

import org.apache.commons.lang3.RandomUtils;
import org.bson.types.ObjectId;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import br.ufsc.labsec.fraudclassifier.classification.EntradaClassifier;
import br.ufsc.labsec.fraudclassifier.generation.BooleanGenerator;
import br.ufsc.labsec.fraudclassifier.generation.CalendarGenerator;
import br.ufsc.labsec.fraudclassifier.model.Entrada;
import br.ufsc.labsec.fraudclassifier.model.OpcaoCidade;
import br.ufsc.labsec.fraudclassifier.model.commons.Classificacao;
import br.ufsc.labsec.fraudclassifier.model.commons.SolicitacaoEmissao;
import br.ufsc.labsec.fraudclassifier.persistence.repository.EntradaRepository;
import br.ufsc.labsec.fraudclassifier.persistence.repository.
    OpcaoCidadeRepository;
import br.ufsc.labsec.fraudclassifier.service.EntradaService;

@RestController
public class EntradaController {

  @Autowired
  private EntradaService service;

  @Autowired
  private EntradaRepository repository;

  @Autowired
  private EntradaClassifier classifier;

  @RequestMapping(value = "/classify", method = RequestMethod.POST)
  public HttpEntity<?> classify(@RequestBody final SolicitacaoEmissao
      solicitacaoEmissao) throws Exception {
    final Entrada entrada = service.save(solicitacaoEmissao);
    if(entrada == null) {
      return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
    }
```

```
  return new ResponseEntity<>(service.classify(entrada), HttpStatus.OK);
}

@RequestMapping(value = "/classify/{id}", method = RequestMethod.GET)
public HttpEntity<?> classify(@PathVariable final String id) throws Exception
    {
  final Entrada entrada = repository.findOne(new ObjectId(id));
  if(entrada == null) {
    return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
  }
  return new ResponseEntity<>(service.classify(entrada), HttpStatus.OK);
}

@RequestMapping(value = "/classify/{id}", method = RequestMethod.POST)
public HttpEntity<?> classify(@PathVariable final String id, @RequestParam("f
   ") final Boolean fraude) throws Exception {
  final Entrada entrada = repository.findOne(new ObjectId(id));
  if(entrada == null) {
    return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
  }

  final Classificacao classificacao = service.classify(entrada);

  service.setFraude(entrada, fraude);
  return new ResponseEntity<>(classificacao, HttpStatus.OK);
}

@RequestMapping(value = "/evaluate", method = RequestMethod.GET)
@PreAuthorize("hasRole('ADMIN')")
public HttpEntity<?> evaluateModel(@RequestParam(name = "folds", required =
   false, defaultValue = "10") final int folds,
  @RequestParam(name = "arff", required = false, defaultValue = "false")
     final boolean arff) throws Exception {
  return new ResponseEntity<>(classifier.evaluateModel(folds, arff).
     replaceAll("\\n", "<br/>"), HttpStatus.OK);
}

private static final String SIMILARITY_THRESHOLD = "0.4";

private static final String GENERATION_COUNT = "50000";

private static final String ID_EXISTENCE_ODDS = "20";
private static final String CNPJ_EXISTENCE_ODDS = "20";
private static final String OTHER_DOCUMENTS_EXISTENCE_ODDS = "10";

public static boolean randomFraude(final Entrada entrada, final double
   threshold) {
  if(entrada.getSimilaridadeFotos() < threshold || entrada.
     getSimilaridadeAssinaturas() < threshold || entrada.
     getSimilaridadeDigitais() < threshold) {
    return true;
  }
  if(Boolean.FALSE.equals(entrada.isDataExpedicaoDiaUtilRg()) || Boolean.
     FALSE.equals(entrada.isDataExpedicaoDiaUtilCnh())
    || Boolean.FALSE.equals(entrada.isDataExpedicaoDiaUtilCne()) || Boolean.
       FALSE.equals(entrada.isDataExpedicaoDiaUtilPassaporte())
```

```
      || Boolean.FALSE.equals(entrada.isDataExpedicaoDiaUtilCtps()) || Boolean.
         FALSE.equals(entrada.isDataExpedicaoDiaUtilCCProfissional())
      || Boolean.FALSE.equals(entrada.isDataExpedicaoDiaUtilCnpj()) || Boolean.
         FALSE.equals(entrada.isDataExpedicaoDiaUtilCertidao())) {
      return true;
    }
    return RandomUtils.nextInt(0, 20) == 0;
  }

}
```

2.10 Ignore.java

```
package br.ufsc.labsec.fraudclassifier.classification.annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Ignore {}
```

2.11 Label.java

```
package br.ufsc.labsec.fraudclassifier.classification.annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Label {}
```

2.12 GeneratesFraudProbability.java

```
package br.ufsc.labsec.fraudclassifier.classification.annotation;

import static java.lang.annotation.ElementType.FIELD;

import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface GeneratesFraudProbability {}
```

2.13 DiaNaoUtil.java

```
package br.ufsc.labsec.fraudclassifier.model;

import java.util.Date;
```

```
import org.bson.types.ObjectId;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.index.Indexed;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "dia_nao_util")
public class DiaNaoUtil {

  @Id
  private ObjectId id;

  @Indexed
  private Date data;

  private String nome;

  public ObjectId getId() {
    return id;
  }

  public void setId(final ObjectId id) {
    this.id = id;
  }

  public Date getData() {
    return data;
  }

  public void setData(final Date data) {
    this.data = data;
  }

  public String getNome() {
    return nome;
  }

  public void setNome(final String nome) {
    this.nome = nome;
  }

}
```

2.14 DiaNaoUtilDAO.java

```
package br.ufsc.labsec.fraudclassifier.persistence.dao;

import java.util.Calendar;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.model.DiaNaoUtil;
```

```
@Component
public class DiaNaoUtilDAO {

  @Autowired
  private MongoTemplate mongoTemplate;

  public boolean existsForDate(final Calendar date) {
    final Calendar start = (Calendar) date.clone();
    start.set(Calendar.HOUR_OF_DAY, 0);
    start.set(Calendar.MINUTE, 0);
    start.set(Calendar.SECOND, 0);

    final Calendar end = (Calendar) date.clone();
    end.set(Calendar.HOUR_OF_DAY, 23);
    end.set(Calendar.MINUTE, 59);
    end.set(Calendar.SECOND, 59);

    return mongoTemplate.exists(new Query(
      Criteria.where("data")
        .gt(start)
        .lt(end)),
      DiaNaoUtil.class,
      "dia_nao_util");
  }

}
```

2.15 DiaNaoUtilRepository.java

```
package br.ufsc.labsec.fraudclassifier.persistence.repository;

import org.bson.types.ObjectId;
import org.springframework.data.mongodb.repository.MongoRepository;

import br.ufsc.labsec.fraudclassifier.model.DiaNaoUtil;

public interface DiaNaoUtilRepository extends MongoRepository<DiaNaoUtil,
   ObjectId> {}
```

2.16 AbstractAttribute.java

```
package br.ufsc.labsec.fraudclassifier.classification;

import weka.core.Attribute;

public class AbstractAtribute extends Attribute {

  private static final long serialVersionUID = 1L;

  public AbstractAtribute(final String attributeName) {
    super(attributeName);
  }

}
```

2.17 BlackBoxComparator.java

```
package br.ufsc.labsec.fraudclassifier.classification.utils;

import org.apache.commons.lang3.RandomUtils;

public interface BlackBoxComparator<T> {

  default Double compare(final T a, final T b) {
    if(RandomUtils.nextInt(0, 20) > 0) {
      return RandomUtils.nextDouble(0.6, 1);
    }
    return RandomUtils.nextDouble(0, 0.6);
  }

}
```

2.18 WrittenSignatureComparator.java

```
package br.ufsc.labsec.fraudclassifier.classification.comparation;

import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.classification.utils.BlackBoxComparator;

@Component
public class WrittenSignatureComparator implements BlackBoxComparator<byte[]>
    {}
```

2.19 BiometricSignatureComparator.java

```
package br.ufsc.labsec.fraudclassifier.classification.comparation;

import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.classification.utils.BlackBoxComparator;

@Component
public class BiometricSignatureComparator implements BlackBoxComparator<byte[]>
    {}
```

2.20 FacesComparator.java

```
package br.ufsc.labsec.fraudclassifier.classification.comparation;

import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.classification.utils.BlackBoxComparator;

@Component
public class FacesComparator implements BlackBoxComparator<byte[]> {}
```

2.21 BusinessDayVerifier.java

```
package br.ufsc.labsec.fraudclassifier.classification.comparation;
```

```
import java.util.Calendar;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import br.ufsc.labsec.fraudclassifier.persistence.dao.DiaNaoUtilDAO;

@Component
public class BusinessDayVerifier {

  @Autowired
  private DiaNaoUtilDAO diaNaoUtilDAO;

  public boolean isBusinessDay(final Calendar calendar) {
    switch(calendar.get(Calendar.DAY_OF_WEEK)) {
      case Calendar.SATURDAY:
      case Calendar.SUNDAY:
        return false;

      default:
        return !diaNaoUtilDAO.existsForDate(calendar);
    }
  }

}
```

3 fraud-classifier-proxy

3.1 App.java

```
package br.ufsc.labsec.fraudclassifier.proxy;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.support.SpringBootServletInitializer;

@SpringBootApplication
public class App extends SpringBootServletInitializer {

  public static void main(final String[] args) {
    SpringApplication.run(App.class, args);
  }

}
```

3.2 WebConfig.java

```
package br.ufsc.labsec.fraudclassifier.proxy.client.config;

import java.util.List;

import org.apache.commons.lang3.time.DateFormatUtils;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.http.MediaType;
import org.springframework.http.converter.HttpMessageConverter;
import org.springframework.http.converter.json.Jackson2ObjectMapperBuilder;
```

```
import org.springframework.http.converter.json.
    MappingJackson2HttpMessageConverter;
import org.springframework.web.servlet.config.annotation.
    ContentNegotiationConfigurer;
import org.springframework.web.servlet.config.annotation.
    WebMvcConfigurerAdapter;

@SpringBootConfiguration
public class WebConfig extends WebMvcConfigurerAdapter {

  @Override
  public void configureContentNegotiation(final ContentNegotiationConfigurer c)
      {
    c.defaultContentType(MediaType.APPLICATION_JSON);
  }

  @Override
  public void configureMessageConverters(final List<HttpMessageConverter<?>>
     converters) {
    converters.add(new MappingJackson2HttpMessageConverter(new
       Jackson2ObjectMapperBuilder()
      .simpleDateFormat(DateFormatUtils.ISO_8601_EXTENDED_DATE_FORMAT.
         getPattern())
      .build()));
  }

}
```

3.3 RestOperationsConfig.java

```
package br.ufsc.labsec.fraudclassifier.proxy.client.config;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;

import org.assertj.core.util.Preconditions;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.web.client.RestOperations;
import org.springframework.web.client.RestTemplate;

import br.ufsc.labsec.fraudclassifier.proxy.client.SSLSecuredRestOperations;

@SpringBootConfiguration
@PropertySource({ "classpath:application.properties" })
public class RestOperationsConfig {
```

```
  @Autowired
  private Environment env;

  @Bean
  public RestOperations restOperations() {
    try {
      final KeyStore keyStore = KeyStore.getInstance(env.getProperty("server.
          ssl.key-store-type", KeyStore.getDefaultType()));
      final char[] password = env.getProperty("server.ssl.key-store-password",
          "").toCharArray();
      keyStore.load(new FileInputStream(new File(Preconditions.checkNotNull(env
          .getProperty("server.ssl.key-store")))), password);
      return new SSLSecuredRestOperations(keyStore, password);
    } catch(KeyManagementException | UnrecoverableKeyException |
       NoSuchAlgorithmException
      | KeyStoreException | CertificateException | IOException |
         NullPointerException e) {
      return new RestTemplate();
    }
  }

}
```

3.4 SSLSecuredRestOperations.java

```
package br.ufsc.labsec.fraudclassifier.proxy.client;

import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.UnrecoverableKeyException;

import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.springframework.http.client.HttpComponentsClientHttpRequestFactory;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.context.annotation.ApplicationScope;

@ApplicationScope
public class SSLSecuredRestOperations extends RestTemplate {

  public SSLSecuredRestOperations(final KeyStore keyStore, final char[]
      password)
    throws KeyManagementException, UnrecoverableKeyException,
       NoSuchAlgorithmException, KeyStoreException {
    super(new HttpComponentsClientHttpRequestFactory(
      HttpClients.custom().setSSLSocketFactory(
        new SSLConnectionSocketFactory(
          new SSLContextBuilder()
            .loadTrustMaterial(null, new TrustSelfSignedStrategy())
            .loadKeyMaterial(keyStore, password).build()))
        .build()));
```

```
  }

}
```

3.5 Log.java

```
package br.ufsc.labsec.fraudclassifier.proxy.model;

import org.bson.types.ObjectId;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "log")
public class Log {

  @Id
  private ObjectId id;

  private ObjectId idEntrada;
  private Long idUsuario;
  private boolean aprovada;
  private Double probabilidadeFraude;
  private Double probabilidadeAutentica;

  public ObjectId getId() {
    return id;
  }

  public void setId(final ObjectId id) {
    this.id = id;
  }

  public ObjectId getIdEntrada() {
    return idEntrada;
  }

  public void setIdEntrada(final ObjectId idEntrada) {
    this.idEntrada = idEntrada;
  }

  public Long getIdUsuario() {
    return idUsuario;
  }

  public void setIdUsuario(final Long idUsuario) {
    this.idUsuario = idUsuario;
  }

  public boolean isAprovada() {
    return aprovada;
  }

  public void setAprovada(final boolean aprovada) {
    this.aprovada = aprovada;
  }
```

```
  public Double getProbabilidadeFraude() {
    return probabilidadeFraude;
  }

  public void setProbabilidadeFraude(final Double probabilidadeFraude) {
    this.probabilidadeFraude = probabilidadeFraude;
  }

  public Double getProbabilidadeAutentica() {
    return probabilidadeAutentica;
  }

  public void setProbabilidadeAutentica(final Double probabilidadeAutentica) {
    this.probabilidadeAutentica = probabilidadeAutentica;
  }

}
```

3.6 LogRepository.java

```
package br.ufsc.labsec.fraudclassifier.proxy.persistence.repository;

import org.bson.types.ObjectId;
import org.springframework.data.mongodb.repository.MongoRepository;

import br.ufsc.labsec.fraudclassifier.proxy.model.Log;

public interface LogRepository extends MongoRepository<Log, ObjectId> {}
```

3.7 ProxyService.java

```
package br.ufsc.labsec.fraudclassifier.proxy.service;

import org.bson.types.ObjectId;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestOperations;

import br.ufsc.labsec.fraudclassifier.model.commons.Classificacao;
import br.ufsc.labsec.fraudclassifier.model.commons.SolicitacaoEmissao;
import br.ufsc.labsec.fraudclassifier.proxy.model.Log;
import br.ufsc.labsec.fraudclassifier.proxy.persistence.repository.
   LogRepository;

@Service
public class ProxyService {

  @Value("${fraud-classifier.host}")
  private String baseUrl;

  @Autowired
  private RestOperations rest;

  @Autowired
  private LogRepository logRepository;
```

```
  private String url(final String action) {
    return baseUrl.concat(action);
  }

  public Classificacao classify(final SolicitacaoEmissao solicitacaoEmissao) {
    return rest.postForObject(url("/classify"), solicitacaoEmissao,
        Classificacao.class);
  }

  public void classify(final String idEntrada, final Long idUsuario, final
      boolean f) {
    final Log log = new Log();
    log.setIdEntrada(new ObjectId(idEntrada));
    log.setIdUsuario(idUsuario);
    log.setAprovada(!f);

    final Classificacao classificacao = rest.postForObject(url("/classify/{
        idEntrada}?f={f}"), null, Classificacao.class, idEntrada, f);

    log.setProbabilidadeFraude(classificacao.getProbabilidadeFraude());
    log.setProbabilidadeAutentica(classificacao.getProbabilidadeAutentica());

    logRepository.save(log);
  }

}
```

3.8 ProxyController.java

```
package br.ufsc.labsec.fraudclassifier.proxy.endpoint;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import br.ufsc.labsec.fraudclassifier.model.commons.SolicitacaoEmissao;
import br.ufsc.labsec.fraudclassifier.proxy.service.ProxyService;

@RestController
public class ProxyController {

  @Autowired
  private ProxyService service;

  @RequestMapping(value = "/classify", method = RequestMethod.POST)
  public HttpEntity<?> classify(@RequestBody final SolicitacaoEmissao
      solicitacaoEmissao) {
    return new ResponseEntity<>(service.classify(solicitacaoEmissao),
```

```
        HttpStatus.OK);
    }

    @RequestMapping(value = "/classify/{idEntrada}", method = RequestMethod.PUT)
    public HttpEntity<?> update(@PathVariable final String idEntrada,
       @RequestBody final Long idUsuario, @RequestParam final boolean aprovada) {
      service.classify(idEntrada, idUsuario, !aprovada);
      return new ResponseEntity<>(HttpStatus.OK);
    }

}
```

ANEXO D – Artigo

Proposta de um Modelo para Detecção de Fraudes na Emissão de Certificados Digitais na ICP-Brasil

Julia Baldissera, Raphael Schwinden da Silveira

Universidade Federal de Santa Catarina (UFSC)
Departamento de Informática e Estatístic
Campus Universitário – Florianópolis – Brazil

{juhbaldissera,raphass22}@gmail.com

Abstract. *Digital signature and certification are efficient mechanisms to ensure authenticity, integrity and confidentiality of electronic documents. The digital certificate issuing entities already have, in their issuing flow, verification of the data related to the request to ensure that it is not a fraud. This process, however, is done manually and is therefore subject to failures due to human error. Because of this, it is proposed a fraud detection model that makes use of Data Mining techniques to be inserted in the flow of digital certificates issuance, so that the process becomes quicker and more accurate. A data classification model was implemented and tested in the flow of a fictitious system, confirming its feasibility and contribution with the agility of the process.*

Resumo. *A assinatura e certificação digital são mecanismos eficientes para garantia de autenticidade, integridade e confidencialidade de documentos eletrônicos. Os órgãos emissores de certificados digitais têm, em seu fluxo de emissão, a verificação dos dados referentes ao pedido para garantir de que não se trate de uma fraude. Este processo, no entanto, é feito manualmente e, por isso, está sujeito a falhas por erro humano. Por conta disto, é proposto um modelo de detecção de fraudes que use técnicas de Data Mining a ser inserido neste fluxo, tornando-o mais preciso e mais ágil. Um modelo de classificação de dados foi implementado e testado no fluxo de um sistema fictício, confirmando sua viabilidade e contribuição com a agilidade do processo.*

1. Introdução

Com o grande avanço da tecnologia no cotidiano das pessoas, muitas ações e transações antes efetuadas presencialmente, passam a ser feitas no meio eletrônico pela internet. Tornou-se possível fazer declarações para a receita federal, emitir notas fiscais, gerar declarações de serviços médicos, fornecer dados para o Sistema de Recolhimento do FGTS (SEFIP), fornecer informações à previdência social, entre outras tarefas (CERTISIGN, 2017).

A assinatura e certificação digital aparecem nesse cenário para facilitar e garantir a autenticidade, integridade e confidencialidade desses documentos eletrônicos (ADAMS; LOYD, 2003). Pessoas e organizações podem criar o certificado em uma

autoridade confiável, e este servirá como uma identidade, pois contém informações do proprietário e servirá para efetuar as assinaturas dos documentos.

O modelo adotado no Brasil para gerenciar certificados digitais é a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil). Composta basicamente por Autoridades Certificadoras (ACs) e Autoridades de Registro (ARs), trabalhando em conformidade com as diretrizes e normas técnicas definidas pela AC Raiz (ITI, 2017b).

Apesar de todas as fiscalizações e especificações técnicas existentes, são noticiados vários casos de fraudes e incidentes com certificados digitais, como o grande esquema de saques irregulares de FGTS (Brasil, 2017a). As consequências destes incidentes podem ser graves, já que documentos eletrônicos são utilizados nas mais diversas áreas da sociedade.

Dito isto, este trabalho apresenta uma proposta de modelo para detecção de fraudes na emissão de certificados digitais na ICP-Brasil. Este modelo é baseado em descoberta de conhecimento usando técnicas de *Data Mining* em cima dos dados existentes. Assim podem ser identificadas irregularidades, que comumente passam despercebidas pelos indivíduos envolvidos.

2. Motivação

O Portal CD Brasil (BRASIL, 2017b) relatou a emissão de mais de 14 milhões de certificados digitais pelo Instituto Nacional de Tecnologia da Informação (ITI) até 2015, sendo que, destes, mais de 3 milhões de certificados foram emitidos em 2015. O mercado de certificado digital mostra uma clara tendência de crescimento.

Desde Junho de 2016, as autoridades da ICP-Brasil passaram a ser obrigadas a comunicar tentativas de fraude e irregularidades na emissão de certificados digitais com o objetivo de popular um banco de dados do novo sistema de comunicado de tentativa de fraude adotado pelo ITI, que oferece uma Lista Negativa para as ARs (ITI, 2017d). Essa lista contém as informações dos supostos fraudadores, incluindo características físicas, imagem do documento de identificação utilizado, entre outros dados. O Agente de Registro (AGR) fica responsável por fazer pesquisas para validar se a pessoa que está no processo de emissão de certificado digital não está envolvida em tentativas de fraudes anteriores.

O modelo de comunicado de fraudes acima mostra-se operacionalmente demorado, considerando a tendência de crescimento do mercado e o aumento da quantia de informações. Muitas vezes decisões são tomadas por intuição de indivíduos, sem levar em consideração o conhecimento incorporado nestes dados (PEI; HAN; KAMBER, 2012). Aqui entra a necessidade de ferramentas e técnicas poderosas para análise dos dados, as quais *Data Mining* fornece, que possibilitem uma predição de possíveis fraudes, conforme mencionado anteriormente.

3. Proposta

A proposta consiste na definição de um modelo para detecção de fraudes no processo de emissão de certificados na ICP-Brasil, que mostre-se mais eficiente que o modelo atual e que não dependa da intuição de indivíduos humanos. Para a arquitetura desta proposta

e para o modelo de *Data Mining* de classificação da fraude, são considerados os seguintes tipos de fraudadores:

- Fraudador externo: a pessoa que deseja emitir o certificado;
    - A pessoa apresenta o documento de identificação original, mas não é a proprietária do mesmo;
    - A pessoa apresenta o documento de identificação falso.
- Fraudador interno: agente de registro que foi corrompido.
    - O AGR, deliberadamente, não reporta a fraude.

O modelo de *Data Mining* foi definido com base no processo de KDD. KDD é a sigla em inglês para "*Knowledge Discovery in Databases*" ("descoberta de conhecimento em bancos de dados") e é o processo utilizado para identificar padrões novos, válidos, úteis e compreensíveis a partir de conjuntos de dados complexos e grandes. Este processo é iterativo, pois as vezes é necessário voltar a etapas anteriores, e interativo, pois não é um processo isolado, as escolhas certas dependem de cada passo e do domínio da aplicação (MAIMON; ROKACH, 2005). O KDD é dividido em 9 etapas.

A primeira etapa de compreensão do domínio da aplicação e os objetivos contemplou-se ao estudar o processo de emissão de certificados digitais na ICP-Brasil, que permitiu o entendimento dos cenários descritos acima. Com isso, identificou-se que o objetivo do *Data Mining* é predizer se uma nova solicitação de emissão de certificado digital trata-se de uma fraude ou não. Aqui são elencadas algumas formas de identificar as tentativas de fraudes:

- Em ambos os casos do fraudador externo, é possível verificar se:
    - A pessoa na foto do documento de identificação é diferente da pessoa na foto capturada durante a validação presencial;
    - Assinatura do documento de identificação é diferente da assinatura colhida na validação presencial;
    - A digital presente no documento de identificação é diferente da digital colhida na validação presencial.
- Para a verificação de uma fraude no documento, também pode-se verificar se a data de expedição do documento foi em um fim de semana ou feriado.

A segunda etapa do KDD engloba a seleção e criação do conjunto de dados. Para isso foi feito um levantamento de todos os dados disponíveis, tanto do requerente do certificado quanto das Autoridades envolvidas, para posterior seleção dos atributos.

A terceira etapa do KDD refere-se ao pré-processamento e limpeza, onde atributos irrelevantes são retirados. Neste caso, os atributos "Nome do requerente" e "CPF do requerente" foram descartados.

Na quarta etapa do KDD é feita a transformação dos atributos para melhorá-los: foram definidas medidas de similaridade entre as imagens do requerente, similaridade entre assinatura digitalizada e a assinatura colhida, assim como a similaridade das digitais; a data de nascimento é convertida em um número que será utilizado para a criação de intervalos relevantes pelo algoritmo de *Data Mining* (número de milissegundos de diferença da data em relação ao dia 1º de janeiro de 1970); as datas de emissões dos documentos foram convertidas em valores lógicos indicando se são dias

úteis ou não; foram gerados índices de ocorrências de fraudes em relação ao total de ocorrências para a cidade e UF de residência do requerente, bem como para a razão social da AC e da AR. Logo, o modelo de *Data Mining* utilizou os seguintes atributos:

- Similaridade da foto do documento de identificação com a foto capturada;
- Similaridade da assinatura do documento de identificação com a assinatura colhida;
- Similaridade da digital do documento de identificação com a digital colhida;
- Data de nascimento numérica;
- Data de emissão do RG não é um dia útil;
- Data de emissão do CNH não é um dia útil;
- Data de emissão CNE não é um dia útil;
- Data de emissão Passaporte Brasileiro não é um dia útil;
- Data de emissão do CNPJ não é um dia útil;
- Data de emissão da Carteira do Conselho Profissional não é um dia útil;
- Data de emissão da Certidão de Nascimento não é um dia útil;
- Data de emissão da Carteira de Trabalho não é um dia útil;
- Índice de ocorrência de fraudes na cidade de residência;
- Índice de ocorrência de fraudes no estado de residência;
- Índice de ocorrência de fraudes pela razão social da AC;
- Índice de ocorrência de fraudes pela razão social da AR.

Durante a quinta etapa do KDD, foi escolhida a tarefa de *Data Mining* apropriada para o projeto, levando em conta os objetivos definidos. A escolha foi a tarefa de classificação, por ser uma das técnicas mais utilizadas quando deseja-se predizer a classe de novos registros.

Após a escolha da tarefa de *Data Mining*, na sexta etapa, foi decidido o algoritmo a ser utilizado. Foram escolhidos algoritmos de árvores de decisão e redes bayesianas para a simulação. Redes neurais não foram consideradas devido à baixa interpretabilidade e demora para efetuar o treinamento. Todos os algoritmos de classificação consideram registros rotulados com a classe, logo, pode-se afirmar que o aprendizado realizado é supervisionado.

O fluxo do processo de detecção de fraudes foi inserido no fluxo atual de emissão de certificado, logo após a submissão dos dados da validação presencial e antes da emissão do certificado em si. Essa adaptação pode ser observada na figura 1 a seguir:

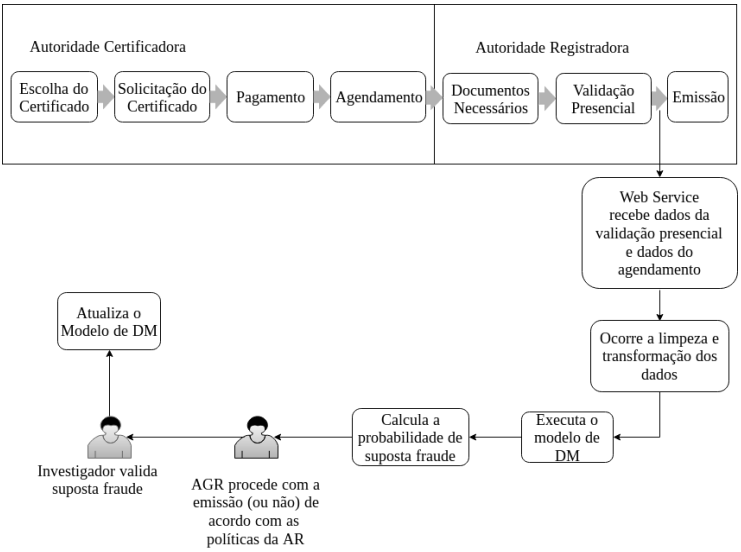

Figura 1. Fluxo da proposta para predição de fraudes

Após a especificação do fluxo de atividades acima, é necessário definir a forma de integrar este modelo com os sistemas das Autoridades já existentes. O ITI não exige que as Autoridades utilizem softwares específicos para o gerenciamento das atividades que envolvem certificação digital. Com base nisso, foi definida uma proposta em que múltiplas Autoridades de Registro se comunicam com o Sistema de Detecção de Fraude (SDF) através de *proxies*. A arquitetura é representada na figura 2, abaixo:

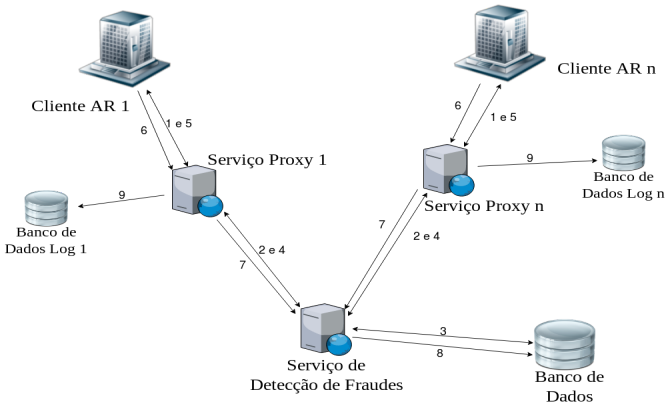

Figura 2. Arquitetura da proposta para predição de fraudes

Os números representados na imagem, são as seguintes ações:

1. A AR envia os dados relacionados ao Agendamento e à validação presencial para o Proxy
2. O Proxy encaminha os dados para o SDF
3. O SDF persiste a entrada não classificada e verifica qual modelo de classificação utilizar:
    a. Caso o modelo não tenha sofrido atualização, o modelo mais recente é utilizado.
    b. Caso o modelo tenha sofrido atualização, um novo modelo é gerado a partir dos dados já classificados;
4. O SDF informa ao Proxy a probabilidade de ser Fraude;
5. O Proxy encaminha à AR a probabilidade;
6. A AR envia o identificador da ocorrência, juntamente com um indicador se esta foi considerada fraude ou legítima ao Proxy;
7. O Proxy encaminha os dados para o SDF;
8. O SDF atualiza a entrada no Banco de Dados desta ocorrência, agora classificada, e sinaliza o modelo como desatualizado (a atualização do modelo de classificação ocorre diariamente à 1h da manhã);
9. O Proxy armazena numa base de dados de registros de operações o identificador do usuário, o indicador se foi considerada fraude ou não, o identificador da entrada e sua probabilidade de fraude.

Para esta arquitetura, foram desenvolvidos 3 projetos: *fraud-classifier*, *fraud-classifier-proxy* e *fraud-detector*. Todos foram construídos utilizando a linguagem de programação Java, *Java Persistence API* (JPA), o motor de banco de dados MongoDB, o *framework* Spring, a ferramenta Apache Maven para gerenciamento de dependências, e o software Git para controle de versão e gerenciamento de alterações de código.

O projeto *fraud-classifier* representa o módulo SDF da arquitetura e consiste de um webservice para classificação das solicitações de emissão de certificados digitais. Um web service é uma Application Programming Interface (API) que, segundo Pautasso (2009), permite comunicação entre aplicações diferentes e disponibiliza seus recursos na rede. O projeto possui os seguintes serviços:

- Calcula a distribuição das classes de uma nova entrada
- Classifica uma determinada entrada

O projeto *fraud-classifier-proxy* representa o módulo Proxy da arquitetura e tem como principal função realizar a ponte de comunicação entre o sistema da AR e o SDF utilizando a autenticação mútua. Este também é um *web service* com a arquitetura REST. O projeto possui os seguintes serviços:

- Calcula a distribuição das classes de uma nova entrada

- Classifica uma determinada entrada
- Visualizar estatísticas do classificador

Uma vez que este projeto tem o objetivo de ser uma ponte de comunicação segura entre o sistema da AR e o SDF, o ponto mais importante deste módulo é o uso de chaves que serão utilizadas na autenticação mútua. Stallings (2008) descreve Autenticação Mútua como um protocolo de autenticação que dá garantia de identidade a ambas as partes da comunicação.Para isto, foram gerados certificados digitais para o SDF e para o Proxy que "confiam" um no outro.

De maneira semelhante ao SDF, as operações são registradas, entretanto, não no arquivo de registros do servidor de aplicação, mas num banco de dados. Uma vez que o Proxy executa em um ambiente interno à AR, é possível armazenar dados mais sensíveis como os identificadores dos usuários que realizaram determinadas ações. Desta forma, numa auditoria, é possível identificar qual usuário aprovou a emissão de um certificado para uma solicitação de autenticidade duvidosa, por exemplo.

4. Simulação

No primeiro momento, foi necessário criar uma base de dados fictícia para treinar os modelos. Após isso, realizou-se a simulação do fluxo de uma emissão de certificado digital, incluindo a adaptação da proposta. Para isso foram criados dados fictícios com o objetivo de simular a execução do modelo de classificação. Phua et al. (2010) comenta em seu artigo, que esta é uma alternativa válida quando se trata de fraudes, pela dificuldade de obter dados reais.

Após a criação do conjunto de dados, é necessário mostrar o funcionamento dos componentes em meio a um fluxo que simule as operações reais de emissão de certificados digitais para prova de conceito da proposta. O projeto *fraud-detector* representa o sistema utilizado pela AR e foi desenvolvido com este propósito. Foi desenvolvido um sistema que simule o cadastro e a aprovação (ou não) de solicitações de emissão de certificados digitais.

O projeto simula o uso por usuários com diferentes níveis de acesso, onde usuários de atendimento têm acesso a funções de cadastro e consulta de agendamentos, enquanto agentes têm acesso, também, às funcionalidades de aprovação (ou não) das solicitações de emissão de certificados fazendo uso do sistema de classificação para auxílio na detecção de fraudes.

Cada AR pode ter uma política de como proceder de acordo com o valor de probabilidade de fraude, mas caso seja alta, não é recomendado emitir o certificado. Dessa forma pode-se seguir com os processos definidos pelo ITI para investigar a irregularidade. A figura 3 a seguir mostra que, após o envio de todos os dados do solicitante do certificado, o serviço retorna a probabilidade de ser fraude. Dessa forma é possível identificar um possível fraudador externo. Nessa mesma interface foi feita a opção de aprovar ou reprovar a fraude.

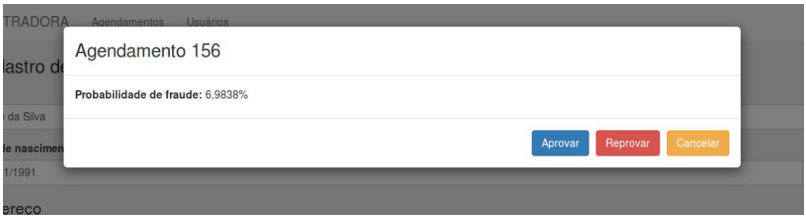

Figura 3. Resultado da classificação feita pelo SDF

Para identificar um possível fraudador interno, pode-se utilizar os dados de log que estão sendo persistidos no banco de dados através do projeto fraud-classifier-proxy. Com a consulta, apresentada na Figura 4 abaixo, são retornados os identificadores dos usuários que aprovaram a emissão de certificado para pessoas que obtiveram uma probabilidade de fraude maior que 50%.

Figura 4. Consulta ao banco de dados de log para identificar fraudador interno

7. Conclusão

O mercado de certificação tem uma clara tendência de crescimento, e os dados gerados sobre as emissões podem revelar irregularidades, que muitas vezes passam despercebidas pelos indivíduos. Já existe um processo de comunicação de irregularidades dentro da ICP-Brasil, mas este mostra-se demorado e mais suscetível a erros, já que considera a intuição dos indivíduos e não leva em conta o conhecimento incorporado nesses dados.

Com um estudo mais detalhado sobre a emissão de certificados digitais, foi possível identificar dois tipos de fraudadores: o fraudador interno, que foi corrompido e

deixa a fraude acontecer, e o fraudador externo se passando por solicitante do certificado. Dentro deste contexto, foram executadas as etapas do processo de KDD, e identificados os dados envolvidos neste processo, e qual etapa é a mais adequada para executar um serviço de detecção de fraudes, neste caso, a etapa de validação presencial.

Como resultado do KDD, obteve-se um modelo de *Data Mining* para classificação de novas entradas, que informa ao agente de registro a probabilidade de ser fraude. Para este trabalho, foi optado por aprendizado de máquina supervisionado, utilizando modelos de classificação *eager*.

Foram definidos dois modelos para a integração com a ICP-Brasil, dos quais optou-se por implementar o modelo que faz uso de um proxy para comunicação segura, uma vez que este se mostrou de mais fácil implantação por parte da AR. Para tal, foram desenvolvidos os módulos do Serviço de Detecção de Fraudes e o Proxy, bem como manuais de instruções para a AC-Raiz e as ARs.

Comprovou-se a viabilidade da proposta através da integração com um sistema de AR fictício, onde todo o fluxo da proposta foi simulado. O resultado desta simulação foi a detecção de um possível fraudador externo, que obteve alta pontuação para fraude, e a detecção de um fraudador interno, que aceitou emitir um certificado para um possível solicitante fraudador.

Como trabalhos futuros, ficou definido: a validação dos dados relacionados à identidade do requerente oriundos de buscas pelo CPF do requerente em bases de dados confiáveis; a utilização de métodos reais para a obtenção dos valores de similaridades ao invés de valores pseudoaleatórios; utilização de outras técnicas de *Data Mining* com aprendizado não supervisionado; a integração do serviço de detecção na ICP-Brasil juntamente com ACs e ARs, e avaliação dos modelos de *Data Mining* gerados a partir de dados reais.

Referências

ADAMS, C.; LLOYD, S. Understanding PKI: Concepts, Standards, and Deployment Considerations. Boston, EUA: Addison-Wesley, 2003

BRASIL, P. C. O mercado de certificação digital brasileiro deverá ultrapassar R$ 2 bilhões em 2020. 2017. <http://portalcdbrasil.com.br/o-mercado-de-certificacao-digital-brasileiro-devera-ultrapassar-2-bilhoes-em-2020/>. Acessado em 22/03/2017.

CERTISIGN. Indicações e uso do Certificado Digital. 2017. <https://www.certisign.com.br/certificado-digital/indicacao-uso>. Acessado em 22/03/2017.

ITI. Certificação Digital ICP-Brasil. 2017. <http://www.iti.gov.br/certificacao-digital>. Acessado em 04/04/2017.

ITI. Institucional ITI. 2017. <http://www.iti.gov.br/institucional>. Acessado em 03/04/2017.

ITI. O que é ICP-Brasil. 2017.<http://www.iti.gov.br/icp-brasil>. Acessado em 20/03/2017.

ITI. Procedimentos para Identificação do Requerente e Comunicação de Irregularidades no processo de emissão de um Certificado Digital ICP-Brasil. 2017. <http://www.iti.gov.br/images/repositorio/legislacao/documentos-principais/DOC-ICP-05.02_Versao_1.3_Procedimentos_de_Identificao_do_Requerente.pdf>. Acessado em 20/03/2017.

PAUTASSO, C. Restful web service composition with bpel for rest. Data & Knowledge Engineering, Elsevier, v. 68, n. 9, p. 851–866, 2009.

PHUA, C. et al. A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119, 2010

PEI, J.; HAN, J.; KAMBER, M. Data mining: Concepts and techniques. [S.l.]: Elsevier, 2012.

MAIMON, O.; ROKACH, L. Data Mining and Knowledge Discovery Handbook. Boston, EUA: Springer Science+Business Media, Inc., 2005.

STALLINGS, W. Criptografia e segurança de redes: princípios e práticas. [S.l.]: Pearson Prentice Hall, 2008.